National Library of Canada
Canadian Theses Service
Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Services des thèses canadiennes

CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL 339 (r. 86/01)

Canada

National Library of Canada / Bibliothèque nationale du Canada

Ottawa, Canada
K1A 0N4

203

TC –

0-315-23397-4

CANADIAN THESES ON MICROFICHE SERVICE – SERVICE DES THÈSES CANADIENNES SUR MICROFICHE

PERMISION TO MICROFILM – AUTORISATION DE MICROFILMER

• Please print or type – Écrire en lettres moulées ou dactylographier

AUTHOR – AUTEUR

Full Name of Author – Nom complet de l'auteur

CHI-SING RAYMOND MA

Date of Birth – Date de naissance

Oct 31, 1959

Canadian Citizen – Citoyen canadien

☑ Yes / Oui ☐ No / Non

Country of Birth – Lieu de naissance

HONG KONG

Permanent Address – Résidence fixe

854 ERIN PLACE
EDMONTON ALBERTA
T5T 1M6

THESIS – THÈSE

Title of Thesis – Titre de la thèse

STANDARD-CELL ROUTING

Degree for which thesis was presented
Grade pour lequel cette thèse fut présentée

M. Sc

Year this degree conferred
Année d'obtention de ce grade

1985

University – Université

UNIVERSITY OF ALBERTA

Name of Supervisor – Nom du directeur de thèse

DR. J. T. MONCHENKO

AUTHORIZATION – AUTORISATION

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to **microfilm** this thesis and to **lend** or **sell copies of the film.**

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de **microfilmer** cette thèse et de **prêter** ou de **vendre des exemplaires du film.**

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

ATTACH FORM TO THESIS – VEUILLEZ JOINDRE CE FORMULAIRE À LA THÈSE

Signature

Chi Sing Raymond MA

Date

Aug 8, 1985

NL-91 (r. 84/03)

Canada

THE UNIVERSITY OF ALBERTA

STANDARD-CELL ROUTING

by

Chi-Sing Raymond MA

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF Master of Science

Department of Electrical Engineering

EDMONTON, ALBERTA

Fall 1985

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR Chi-Sing Raymond MA

TITLE OF THESIS STANDARD-CELL ROUTING

DEGREE FOR WHICH THESIS WAS PRESENTED Master of Science

YEAR THIS DEGREE GRANTED Fall 1985

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) Chi Sing Raymond Ma

PERMANENT ADDRESS:

854 ERIN PLACE

EDMONTON, ALBERTA

CANADA

T5T 1M6

DATED Aug. 2 1985

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled STANDARD-CELL ROUTING submitted by Chi-Sing Raymond MA in partial fulfilment of the requirements for the degree of Master of Science.

...Jack Snoeyink.....

Supervisor

...Keith A. Stevens.....

.....T.G. Pickard...

Date...Aug. 2, 1985.....

Abstract

This thesis deals with the problem of routing standard-cell ICs in a standard-cell layout system implemented at the University of Alberta. The standard-cell router described in this thesis uses a two-step approach. The two steps are *global routing* and *detailed routing*. *Global routing* determines the channels in which each net should be routed. *Detailed routing*, or *channel routing*, determines the exact wiring paths in the channels after global routing.

The major result presented is a new track-filling standard-cell global routing algorithm. This algorithm uses a track-filling approach whose principle is to fill the channels with wire segments track by track. The objective of the algorithm is to minimize the total channel density. Experimental results show that, compared to a minimum spanning tree global router, the track-filling algorithm routes 15 test circuits with slightly fewer tracks, while using about the same amount of time.

A minor result presented is a gridless channel routing algorithm based on the Greedy Channel Router due to Rivest and Fiduccia.

Acknowledgement

I would like to thank my thesis supervisor, Dr. J.T. Mowchenko, for his constant support and guidance throughout the past two years. I would also like to thank my examination committee members, Dr. K.A. Stromsmoe and Dr. T.A. Marsland, for their comments on my thesis. In addition, I would like to thank fellow-students, N. Shah, J. Savarimuthu, and R. Singh, for their fellowship and many enlightening discussions on VLSI layout, and B. Zylstra for designing some of the test circuits and entering them into the circuit database.

Finally, I am deeply grateful to my parents for their love, understanding and encouragement.

This research was supported by a NSERC Postgraduate Scholarship.

Table of Contents

Chapter Page

1. Introduction 1

2. Standard-Cell Routing 10

2.1 IC Routing in General 10

2.2 Classification of Routing Algorithms 11

2.3 Standard-Cell Routing 13

2.3.1 The Standard-Cell Detailed Routing Problem 14

2.3.2 The Standard-Cell Global Routing Problem ..19

2.4 The U. of A. Standard-Cell System 24

3. Detailed Routing 28

3.1 Original Greedy Channel Router 28

3.1.1 Problem Definition 28

3.1.2 The Greedy Channel Router 29

3.1.3 Advantages and Disadvantages of the Greedy Channel Routing Algorithm 38

3.2 Gridless Greedy Channel Router 39

3.2.1 Problem Definition 39

3.2.2 Extension of the Greedy Channel Router to a Gridless Channel Router 41

3.2.3 The Gridless Greedy Channel Router 48

4. Global Routing 52

4.1 Definitions and Concepts 52

4.2 Problem Definition 57

4.3 A Track-Filling Approach to Standard-Cell Global Routing 60

4.4 The Track-Filling Algorithm 61

4.4.1 Mutually Exclusive Set of Wire Segments ...63

4.4.2 Fullness & Density Estimates 64

4.4.3 Selection of Segments From the Mutually Exclusive Set 66

4.4.4 Global and Local Subnets 72

4.5 Improvement of Initial Solution by Segment Swapping 74

5. Experiments and Evaluation 79

5.1 The Routing Programs and Test Circuits 79

5.2 Investigation of the Parameters 80

5.3 Evaluation of the Algorithm 82

5.4 Some Observations 84

6. Conclusions and Future Research 101

Bibliography 105

Appendix I - Steiner Tree in Graph Problem 109

Appendix II - A Minimum-Spanning-Tree Global Router 110

List of Tables

Table		Page
5.1	Statistics of the Test Circuits	88
5.2	Experimental Results Obtained with Variations of Parameters	89
5.3	Experimental Results Obtained With Track-Filling Algorithm, Tree-Type Algorithm, No Global Routing, and Left-Edge Method	90
5.4	Experimental Results Obtained With 4-Pass Improvement Step	91
5.5	Correlation of ΣDf with $D'*E'/E$, and Increase of Wiring Area	92

List of Figures

Figure		Page
1.1	Standard-Cell Approach	4
1.2	Gate-Array Approach	5
1.3	General-Cell Approach	6
2.1	Manhattan Routing	12
2.2	A Channel	15
2.3	Vertical Constraints in Channel Routing	17
2.4	Vertical Constraint Graph	18
2.5	Cyclic Vertical Constraints	18
2.6	Tree-Type Approach to Standard-Cell Global Routing	22
2.7	Standard-Cell Layout Model	25
2.8	Advantages of Double-Entry Cells	26
3.1	Greedy Channel Routing Algorithm	30
3.2	A Jog	31
3.3	Assign Tracks To Nets At Left Edge	33
3.4	Bring In Top and Bottom Nets	33
3.5	Collapsing Jogs	35
3.6	Choose Collapsing Jogs Algorithm	36
3.7	Reduce Range Of Split Nets	36
3.8	Raise or Lower Nets	37
3.9	Widen Channel If Necessary	37
3.10	Extend To Next Column	38
3.11	Disadvantage of Greedy Channel Router	39
3.12	Problems of Gridless Routing	42
3.13	Column Generation Algorithm	43

Figure		Page
3.14	Gridless Columns	44
3.15	Column Generation	45
3.16	Example of Column Generation	46
3.17	Neighboring Columns Which Share a Terminal	47
3.18	Relaxed Wiring Model	48
3.19	Gridlesss Channel Routing Algorithm	49
3.20	Gridless Collapsing Jogs	50
3.21	Example of Routed Gridless Channel	50
4.1	Net Segments of A Net	53
4.2	Implementation of a Net Segment	54
4.3	Wire Segments of a Net	54
4.4	Classification of Wire Segments	56
4.5	Graph Representation of Net Segments	57
4.6	An Global Routing Problem Example	59
4.7	A Track used in Track-Filling Approach	60
4.8	Track-Filling Algorithm	62
4.9	Three Different Density Estimates	67
4.10	Ranking of Segments in a Mutually Exclusive Set	68
4.11	Managing Local Densities Using Switchable Segments	69
4.12	Essential Segments Must Be Used	70
4.13	Choosing Non-Essential Segment or Non-Switchable Segment	70
4.14	Need of Local Subnets	73
4.15	Local Subnet	74

Figure Page

4.16 Segment Swapping 75

4.17 Four Pass Segment Swapping 76

4.18 Segment Swapped Back and Forth 77

4.19 Purpose of Pass 3 and Pass 4 78

5.1 Checkplot of a Circuit After Routing 93

5.2 log_{10}(number of terminals) vs log_{10}(time) for Track-Filling Algorithm 94

5.3 log_{10}(number of canonical wire segments) vs log_{10}(time) for Track-Filling Algorithm 95

5.4 $log_{10}(E log_2 e)$ vs log_{10}(time) for Tree-Type Algorithm 96

5.5 $log_{10}(E log_2 e)$ vs log_{10}(time) for Track-Filling Algorithm 97

5.6 Plot of the Three Density Estimates During Global Routing 98

5.7 Tracks Obtained With *FMAX*=1.0 98

5.8 Tracks Obtained With *FMAX*=0.7 99

5.9 Tracks Obtained Using Left-Edge Method 99

5.10 Initial and Final Density Profiles 100

5.11 Checkplot of Circuit 4 After Routing 100

6.1 Maximize the Amount of Metal Used in Routing 104

Chapter 1

Introduction

As the technology for fabricating integrated circuits (ICs) improves, the number of transistors that can be put on a chip continues to increase. The level of integration has evolved from large scale integration (LSI) in the early 1970's to very large scale integration (VLSI) in the late 1970's. Chips with approximately 500,000 transistors have already been fabricated and million-transistor chips are not far off. It is obvious that the complexity involved in the design of VLSI chips is enormous. In order to manage this complexity, computer-aided-design (CAD) tools are used in every aspect of the IC design cycle. One part of the IC design process which uses CAD extensively is the layout of the IC.

The *layout* of an IC refers to the process of translating a specification of an integrated circuit into photolithographic masks for fabricating the circuit. ICs are formed on silicon wafers by creating different layers of conducting substances (eg. diffusion, polysilicon, metal) in geometric patterns on the wafer. Electronic devices like transistors are formed by the interaction between overlapping regions on different layers. In order for the fabricated circuit to function properly, certain rules called *design rules* must be observed when the circuit is

being laid out. These rules specify the minimum dimension of the objects on each layer and the minimum separation between different objects on either the same or different layers. For example, a metal wire must be wider than a certain size and two metal wires must be separated by at least a certain distance.

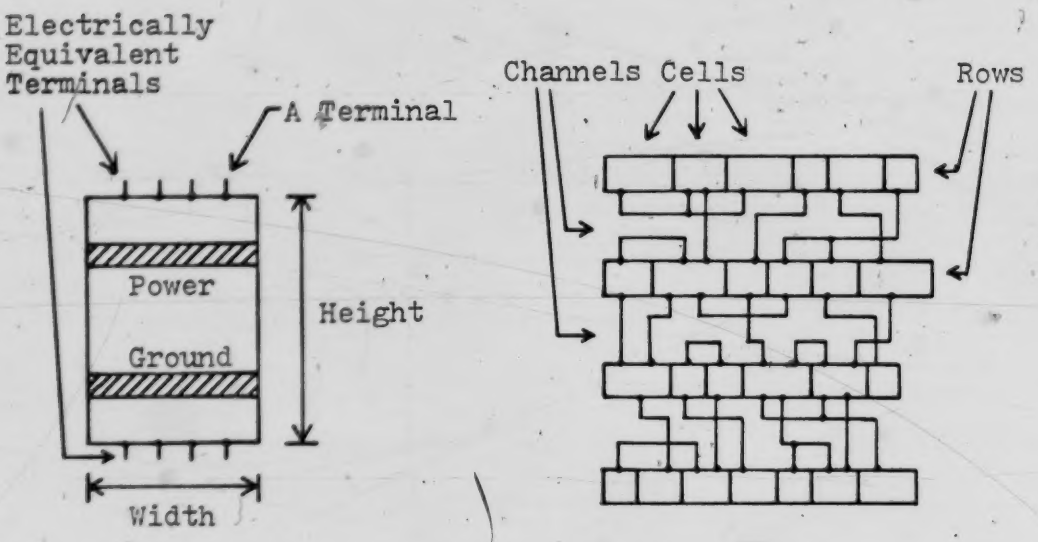
As increasingly complex circuits can be fabricated on a single chip, more and more "customized" ICs, or *custom ICs*, are being designed for specific applications. Custom ICs can be divided into two types: *full custom* and *semi-custom*. *Full custom* ICs are usually designed and laid out manually at the transistor level. The development time and costs for these ICs are very high and therefore, the full custom approach is typically only used for circuits which require stringent performance, minimum chip size, or are produced in high volume. The majority of custom ICs designed are semi-custom ICs. *Semi-custom* ICs are mainly designed and laid out in the so called *cell-based* design style. In *cell-based* design, the term *cell* refers to a functional unit such as a NAND gate, NOR gate, transmission gate, D flip-fop, etc., whose mask description is held in a library. A designer specifies the design in terms of these cells and their interconnections. The layout can be produced by highly automated software so that the designer never has to deal with the mask information. Hence, by using cell-based design, the development time and costs for semi-custom ICs are lower than for full-custom ICs. Another benefit of using

cell-based design is that the chance of design error is reduced.

There are three major cell-based IC design styles: *standard cells*, *gate arrays*, and *general cells*.

In the *standard-cell* (also called *polycell*) approach, the designer is provided with a library of basic functional units called standard cells. The library contains the complete mask layout of the functional units. The layout of a standard cell occupies a rectangular region as shown in Fig. 1.1a. Typically, all cells in a library have the same height but varying widths. A cell's signal I/O terminals lie along its top and/or bottom borders. Early standard cells were designed with terminals on only one border. Standard cells currently in use typically have terminals on both sides of the cell and have the same terminal pattern on the top and bottom. In other words, terminals which are diametrically opposite on the cell are electrically equivalent, as shown in Fig. 1.1a. We will refer to cells with such a terminal configuration as *double-entry* cells. A standard-cell IC is laid out by placing individual cells end-to-end in horizontal rows and routing the terminal interconnections in the *channels* between neighboring rows. Fig. 1.1b shows a typical standard-cell layout.

A *gate-array* (or *master-slice*) is a pre-fabricated IC which consists of a regular array of identical, fully functional logic cells. Each cell is usually a simple gate. The array of cells is uncommitted, i.e., internal



(a) A Standard Cell (b) A Typical Standard-Cell Layout

Fig. 1.1 Standard-Cell Approach

transistors are connected to form the cells, but the cells are not connected to form a circuit. Hence gate-arrays are also called *uncommitted logic arrays* (ULAs). Implementing a gate-array IC, also called customization of a gate-array, consists of fabricating one or more interconnect layers on top of the ULA. These layers connect the pre-fabricated cells to form the circuit. The cells of a gate-array may be arranged in horizontal rows separated by horizontal routing channels (Fig. 1.2a), or in an island-cell style two-dimensional matrix with horizontal and vertical routing channels (Fig. 1.2b), or in butting-cell matrix style which allows interconnections through the cells (Fig. 1.2c).

In the *general-cell* (also called *building block* or *macro-cell*) approach the functional blocks implement more complex functions and are physically larger than the

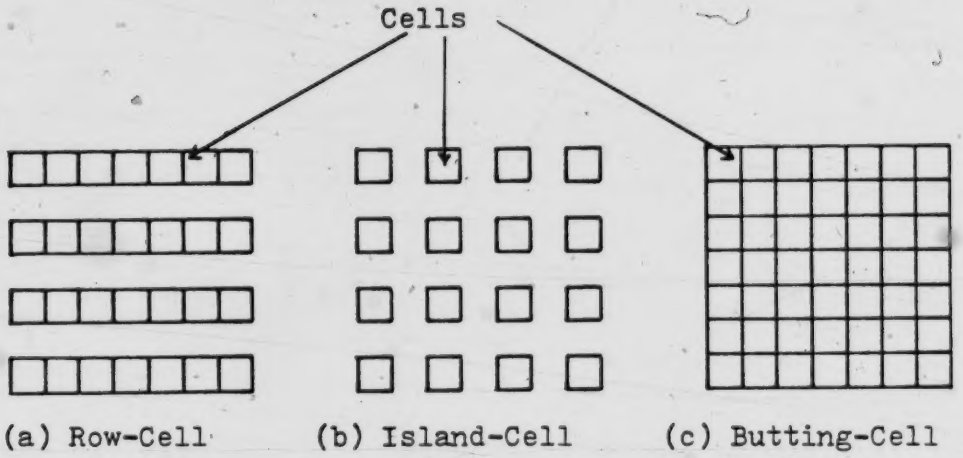
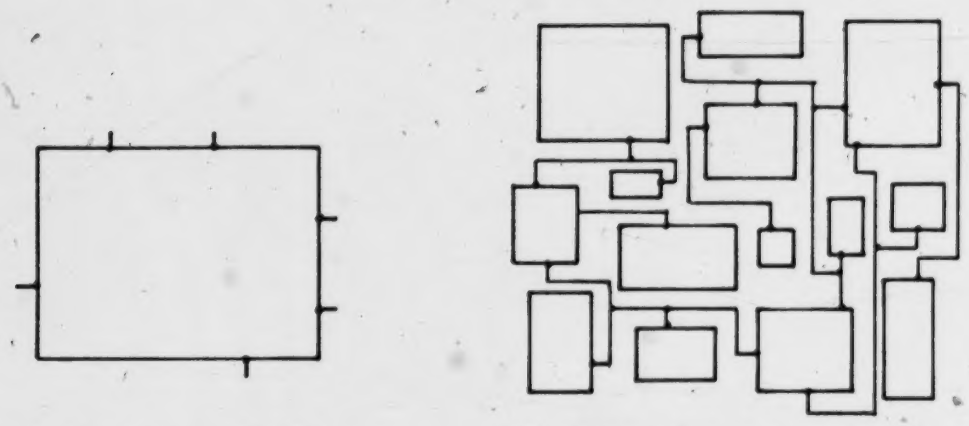


Fig. 1.2 Gate-Array Approach

functional blocks in the standard-cell or gate-array design styles. There are no restrictions on the physical size or shape of a general cell, and a general cell's terminals may be on any side of its perimeter (Fig. 1.3a). Examples of typical general-cell functions are shift registers, arithmetic units, programmable logic arrays (PLAs), read-only-memories (ROMs), etc. A general-cell layout consists of the cells arranged in an arbitrary, but compact, fashion with the interconnects running in the spaces between the cells (Fig. 1.3b).

Each of the cell-based design styles has its advantages and disadvantages. Standard-cell ICs use area more efficiently and perform better than gate-array ICs, but the design costs, manufacturing costs, and turnaround time for gate-array ICs are lower. Both standard cells and gate-arrays are suited to automated layout because of their regular structure. However, the restriction in the shape and



(a) A General Cell (b) A General-Cell Layout

Fig. 1.3 General-Cell Approach

size of standard cells and gate-array cells allows only simple gates to be put into each cell. General cells are suited to complex logic functional blocks because of their arbitrary shape and size. However, automatic layout of general-cell ICs is much more difficult than the other two approaches.

As mentioned above, the process of laying out cell-based ICs can be automated with a software system. Such a system takes a description of the circuit topology, i.e. the cells and their interconnections, and produces mask descriptions for the IC.

The circuit topology is usually captured in a database by one of two methods. In a "schematic capture" system, the designer enters the circuit by drafting the schematic diagram of the circuit on a graphics terminal. In the other method, the designer uses a circuit editor program at an alpha-numeric terminal to enter the circuit by defining the

components of the circuit in terms of the cells from the cell library and their interconnections.

After the circuit description has been entered, the IC is ready to be laid out. The layout process is usually divided into two phases: placement and routing. During *placement*, the physical location of the cells on the IC die are determined by a placement program. The objective of placement is to arrange the cells so that they can be interconnected in a compact fashion in the routing phase. During *routing*, the actual paths of the conductors which interconnect the cell terminals are determined by a routing program. The objective of routing is to connect all signal paths of the circuit, without violating any design rules, in the minimum area. Many placement and routing algorithms are in use. The reader may refer to [Sou81] for a tutorial overview of the circuit layout problem, and the placement and routing algorithms in use. We will concentrate on the routing problem of standard-cell ICs in this thesis.

Routing problems in general are computationally very complex. Routing problems similar to standard-cell routing have been shown to be NP-complete [Gar79, LaP80, Szy85]. This means that there is probably no efficient algorithm that can find the best solution. A common approach to tackle this kind of problem is to use a "heuristic" algorithm which attempts to find a reasonably good solution within an acceptable amount of computation time. Almost all routing algorithms are heuristic algorithms.

A strategy frequently used by routing algorithms is to divide the problem into two subproblems: global routing and detailed routing. During *global routing*, the routing plane is divided into individual routing regions, and rough signal paths through these regions are found. During *detailed routing*, the exact locations of the wiring paths within each individual region are determined. For standard-cell ICs, the routing regions are the channels between the cell rows, and detailed routing of the channels is called *channel routing*.

A standard-cell design system has been developed at the University of Alberta to be used as a research tool for developing and testing layout algorithms. There are four major components in this system: a cell library, a circuit editor, placement programs, and routing programs. This thesis reports the research work done on the routing programs in this system. The objective of this research is to develop improved routers for standard-cell IC design. Early in this work, it was observed that very few standard-cell global routers have been reported in the literature, whereas channel routing has been thoroughly studied in the past [Hash71, Ker73, Deu76, Yos82, Riv82a, Bur83]. Therefore, it was decided that more improvement in standard-cell routing could be obtained through improvements in global routing. The majority of the research work was devoted to developing and experimenting with a new standard-cell global router. A minor component of the research was to modify an existing channel routing algorithm

so that it can be used in our system.

This thesis is organized as follows. In Chapter 2, we review the common approaches used by existing routing systems. We then focus our attention on standard-cell routing, and describe some of the existing methods used for standard-cell global routing and detailed routing. In Chapter 3, we discuss the detailed router used in our system. This router is based on an existing channel router. We present that original channel router and explain how we have modified it for use in our system. In Chapter 4, a new standard-cell global routing algorithm is presented. This algorithm uses an approach different from the approach used by existing routing systems. We have developed and implemented this algorithm to be used in our system. In Chapter 5, we present the experimental results and observations obtained from the experiments on our global router. In Chapter 6, we summarize this research, present conclusions, and offer suggestions for future work.

Chapter 2

Standard-Cell Routing

We begin this chapter by introducing the general IC routing problem and describing several methods commonly used for routing ICs. We then focus on the standard-cell routing problem, which consists of global routing and detailed routing. We will describe the existing methods used for standard-cell global and detailed routing, and point out the drawbacks of these existing methods. Then we describe a standard-cell design system implemented at the University of Alberta. This system uses the global and detailed routing approach to route an IC. A detailed description of the layout model followed by this system is given so that we can formulate the routing problems later.

2.1 IC Routing in General

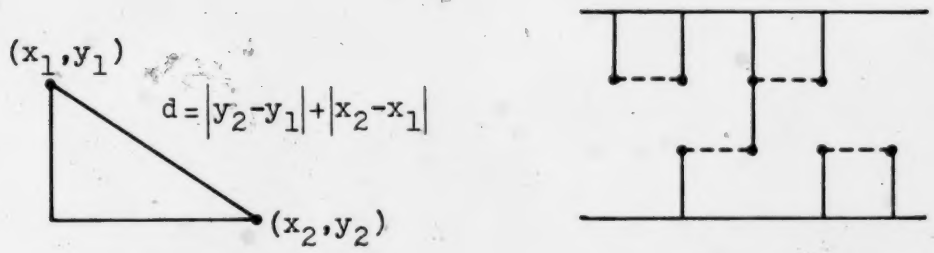
IC routing is the problem of connecting the nets of a circuit with wires in a routing plane. A *net* is a set of cell terminals which are electrically common in a circuit. The connectivity of a circuit is specified by its *net-list* which contains definitions for each net in the circuit. The *routing plane* of an IC is the area on the IC which is reserved for the interconnect wires.

The manner in which the wires are run in the routing plane is governed by a set of rules called the *wiring model*.

The *two-layer Manhattan wiring model* is almost exclusively used for routing standard-cell ICs. In Manhattan wiring, only horizontal and vertical wires are used. Also, rectilinear distance, called *Manhattan distance*, is used instead of Euclidean distance (see Fig. 2.1a). The two-layer Manhattan wiring model assumes that the wires lie on two layers, each reserved for one direction. Two layers are usually used for routing ICs because most fabrication processes only have two layers available for interconnect: metal and polysilicon. Typically, for standard-cell ICs, horizontal wires lie on the metal layer and vertical wires lie on the polysilicon layer. To connect a horizontal wire and a vertical wire, a *contact* is placed at the point where the two wires intersect. A set of connected horizontal and vertical wires is called a *routing path* or a *signal path*. A net is said to be *carried* by a routing path if the path consists of wires which are connected to the net's terminals. In order to be electrically correct, the routing paths for different nets must be disjoint. An example of a routing in the two-layer Manhattan model is shown in Fig. 2.1b.

2.2 Classification of Routing Algorithms

Routing algorithms can be categorized into two classes: search algorithms and channel algorithms. *Search algorithms* consider the whole routing plane at once and search for the final wiring path of each net consecutively. Examples of



(a) Mahanttan Distance

(b) A Two-Layer Manhattan Routing

Fig. 2.1 Manhattan Routing

search algorithms are the Lee router [Lee61] and the line-router [Hig69]. *Channel algorithms* use a two-step approach to routing. The two steps are: global routing and detailed routing. In *global routing* (also called *loose routing*), the routing plane is divided into smaller routing regions and a rough path through these routing regions is computed for each net by a global router. The rough path for a net consists of simply the routing regions which the net passes through, no exact wiring paths are determined yet. Examples of global routers are described in [Che77, Pre79, Riv82b, Kim83, Sec84]. In *detailed routing* (also called *final routing*), the exact paths of the wires in each routing region are determined by a detailed router. We call algorithms which use this two step approach channel algorithms because invariably the routing regions are rectangular regions called *channels*. Detailed routing of the channels is called *channel routing*. Channel routing was first introduced by Hashimoto and Stevens [Hash71] in 1971.

Since then, channel routing has become very popular and widely used for routing ICs. We will consider channel routing in more detail later in this chapter.

The advantage of search algorithms is that they find the shortest path between each pair of terminals on a net and hence produce good routing solutions. However, they require enormous amounts of computer time and storage space to route large circuits. Channel algorithms reduce the time and space requirements by dividing the problem into global and detailed routing. Consequently, channel algorithms are used mainly now.

Besides being categorized as search or channel algorithms, routing algorithms can also be categorized as either *sequential* or *parallel* routers. A *sequential router* orders the nets in a particular way and then routes them one at a time. For example, the Lee and the line-router are sequential. A *parallel router* works on several nets at the same time. Channel routers are usually parallel routers.

2.3 Standard-Cell Routing

In standard-cell design, the problem of routing is to connect the nets with wires in the channels between the cell rows. Standard-cell ICs have been routed mainly with channel routing algorithms. In this section, we will consider the standard-cell global and detailed routing problems. Since the discussion of global routing will make use of terms and concepts that apply to detailed-routing, we will discuss the

standard-cell detailed routing problem before the global routing problem.

2.3.1 The Standard-Cell Detailed Routing Problem

In standard-cell routing, the routing regions are the rectangular channels between the rows. Detailed routing of these channels is equivalent to channel routing. The input to a channel routing problem generally consists of a net-list which specifies the terminals and their interconnections in the channel. The channel routing problem is to determine the exact wiring paths interconnecting the terminals within a channel routing region in the minimum area. Let us define some of the terms and concepts that apply to channel routing in general. These terms and concepts will be used throughout the remainder of this thesis.

A *channel* (Fig. 2.2) is a rectangular region with terminals on two opposite sides of its perimeter. Each side of the perimeter of a channel is called a *channel edge*. Each channel has four channel edges: *top*, *bottom*, *left*, and *right*. Channel routers usually orient a channel so that the terminals are on the top and bottom edges, and the left and bottom edges form the coordinate-axes. Nets may enter and/or leave a channel through the left and/or right edges, but the locations at which they cross these edges are not fixed.

Frequently the positions of terminals and wires in a channel are restricted so that they lie on a grid consisting

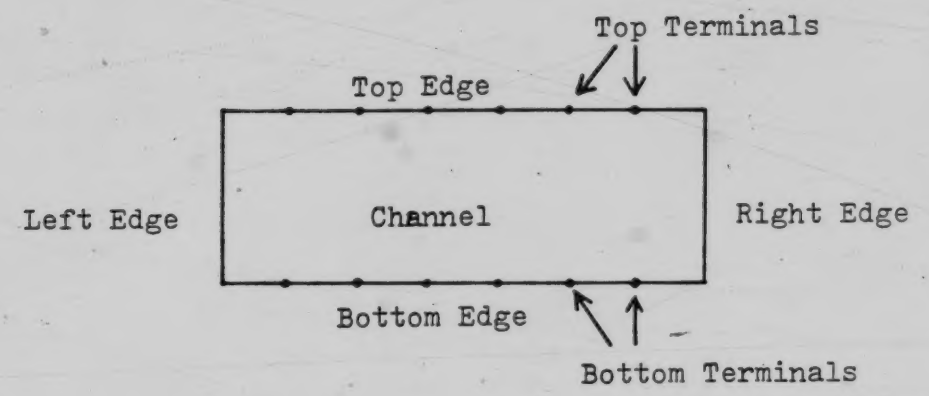


Fig. 2.2 A Channel

of horizontal lines called *tracks* and vertical lines called *columns*. The distance between grid lines usually equals to the minimum separation between adjacent contacts allowed by the design rules. We refer to this distance as the *minimum inter-wire spacing*. Channels on which the grid restriction is imposed are called *gridded channels* and they are routed by *gridded routers*. Channels which do not have this restriction are called *gridless channels* and they are routed by *gridless routers*. Very few truly gridless channel routers have been presented in the literature. Usually, gridless routers do not impose the vertical grid lines, but retain the horizontal grid lines. In other words, horizontal wires must run on horizontal tracks, but vertical wires can run along any abscissa position. We will assume that this is the situation when we refer to gridless routing in the remainder of this thesis.

The *length* of a gridded channel is specified in terms of the number of columns in it. The length of a gridless channel is specified in terms of its real horizontal length. The *width* of a channel, gridded or gridless, is the number of horizontal tracks which is required to route the channel.

The task of a channel router is to route all the nets successfully in minimum area. Since the length of a channel is usually fixed, the objective is to find a routing using the minimum width, i.e., minimum number of tracks.

A mathematical lower bound on the channel width is the density of the channel. Formally, the *local density* at a particular horizontal position x along the channel is defined to be the number of nets whose leftmost terminal at abscissa p and rightmost terminal at abscissa q are such that $p<x\leq q$ or $p\leq x<q$. That is, the local density is the number of nets which must cross, or start, or stop at the horizontal position x in the channel. The *density* of a channel is the maximum of all local densities over the whole channel. Under the two-layer Manhattan wiring model, the channel density is a lower bound on the width of the routed channel. The *span* of a channel is the total length of the channel for which the local density is equal to channel density. The total densities of all the channels in a chip is often used to indicate the overall chip density.

A major problem in channel routing is the existence of *vertical constraints*. Consider Fig. 2.3a. The top terminal belongs to net 1 and the bottom terminal belongs to net 2.

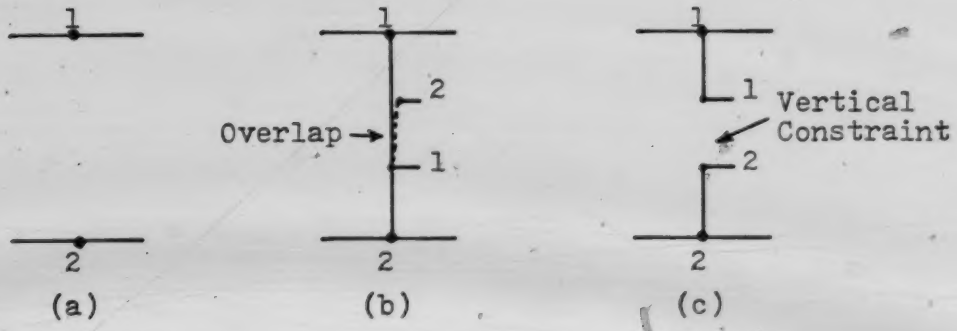
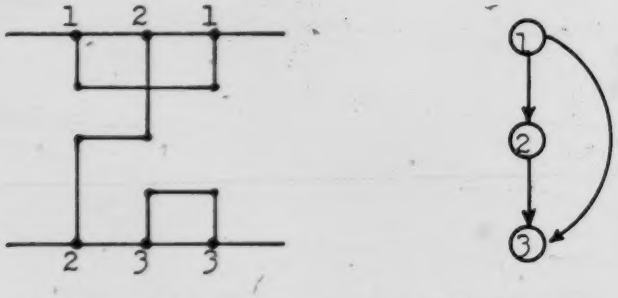


Fig. 2.3 Vertical Constraints in Channel Routing

The horizontal wire for net 2 cannot be placed above the horizontal wire for net 1 because, as shown in Fig. 2.3b, that would result in the two vertical wires overlapping. Hence, there is a vertical constraint that the horizontal wire for net 2 must be below the horizontal wire for net 1, as shown in Fig. 2.3c. We can represent all vertical constraints in a channel by a vertical constraint graph. The *vertical constraint graph* is a directed graph whose vertices represent the nets. Two vertices n_1 and n_2 are connected by an arc from n_1 to n_2 if net n_1 must be placed above net n_2 (Fig. 2.4). It can be shown that if we are restricted to using only one horizontal track for each net, then we cannot route the channel if its vertical constraint graph has cycles. This is because there will always be some vertical wires for different nets which will overlap (Fig. 2.5) [Ker73]. Some cyclic constraints may be avoided by adjusting the placement of the cells. Some channel routers allow the nets to be carried by more than one horizontal wire to make

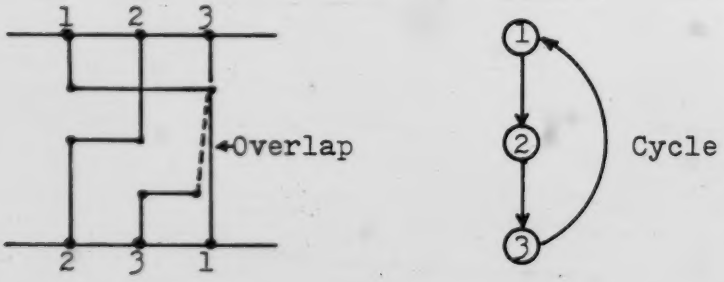


(a) Channel (b) Vertical Constraint Graph

Fig. 2.4 Vertical Constraint Graph

the channel routable even if there are cyclic constraints [Deu76].

Channel routing has been proven to be NP-complete [LaP80, Szy85], therefore almost all channel routers use heuristic algorithms. This is not a serious limitation because most of these routers can solve practical channel routing problems so that the number of tracks is equal to or slightly larger than channel density.



(a) Channel With Cyclic Constraints (b) Vertical Constraint Graph

Fig. 2.5 Cyclic Vertical Constraints

The first channel routing algorithm, due to Hashimoto and Stevens [Hash71], is called the "left edge algorithm". Since the left edge algorithm was introduced, many better channel routing algorithms have been developed [Ker73, Deu76, Yos82, Riv82a, Bur83]. A survey of the channel routers in use today is given in [Bur85].

Almost all of the existing channel routers are gridded. The gridded routing approach imposes constraints on the design of the cells. Each cell must be designed such that the width of a cell and the distance between terminals is a multiple of the grid spacing and this often leads to wastage of silicon area. However, the gridded approach is widely used because it simplifies the channel routing algorithm. The gridless approach allows more compact cells to be designed at the cost of increased complexity of the channel router.

2.3.2 The Standard-Cell Global Routing Problem

The input to a standard-cell global routing problem consists of (1) a placement of standard cells for a circuit following the layout model described above, and (2) the net-list of the circuit. The standard-cell global routing problem, stated in general terms, is to determine the channels in which each net should be routed so that the minimum number of tracks will be used by detailed routing. Since most channel routers can usually route channels at or close to channel density, it is reasonable for a global

router to use the sum of the channel densities for all the channels as the objective function to be minimized.

Most of the global routers discussed in the literature are for general-cell ICs [Pre79, Hass82, Riv82b, Kim83] and gate-array ICs [Che77, Mat82, Tin83, Tsu83, Aos83, Jen84, Wie84]. Very few of them are for standard-cell ICs. Two standard-cell systems which reported using a global router are the TimberWolf Package [Sec84] and the Siemens AVESTA system [Kol77]. Almost all of the existing global routers mentioned above possess similar characteristics. They are all sequential routers which route the nets one at a time. Also, they all operate by finding, for each net, a minimum spanning tree or approximate minimum Steiner tree (see Appendix I) over the terminals of the net, followed by an iterative improvement step. We will refer to this approach the *tree-type approach*.

Let us consider two existing global routers for standard-cell ICs in more detail. The layout model followed by the TimberWolf Package is the same as our layout model. In the TimberWolf Package, each net is modelled by a graph whose vertices represent the net terminals and each edge joins two terminals which are horizontally adjacent. For example, Fig. 2.6b shows the graph for a net in the layout of Fig. 2.6a. The edges are weighted according to their Manhattan length. The global router finds a minimum length spanning tree for each net, then employs an iterative improvement step based on the simulated annealing technique

[Kir82].

The Siemens AVESTA standard-cell system uses a layout model which is different from the one we use. In the AVESTA system, the cells are arranged into double-row blocks separated by horizontal and vertical channels as shown in Fig. 2.6c. The global router of AVESTA uses a graph whose vertices represent the blocks and channel intersections, and whose edges represent the channels as shown in Fig. 2.6d. The global routing of a net is found by computing a Steiner tree on the graph which spans the vertices (which represent blocks) that contain the net's terminals. In computing the Steiner tree, the edges are assigned a weight which is a function of the estimated geometric length of the interconnection and the estimated density of the channel. After all nets have been routed, an iterative improvement step is used to improve the initial solution.

Global routers for row-cell style gate-array ICs can be adapted to standard-cell layout because of the similarities of the two layout styles. The objective of these gate-array global routers is to minimize the maximum channel density because the channel widths in gate-arrays are fixed. Two such global routers are the LTX2 system [Wie84] and the global router reported in [Aos83]. They both model the nets as shown in Figs. 2.6a, and 2.6b and use the sequential tree-type approach. In LTX2, the edges are assigned a weight which is a function of both the geometric length and the estimated density of the interval represented by the edge.

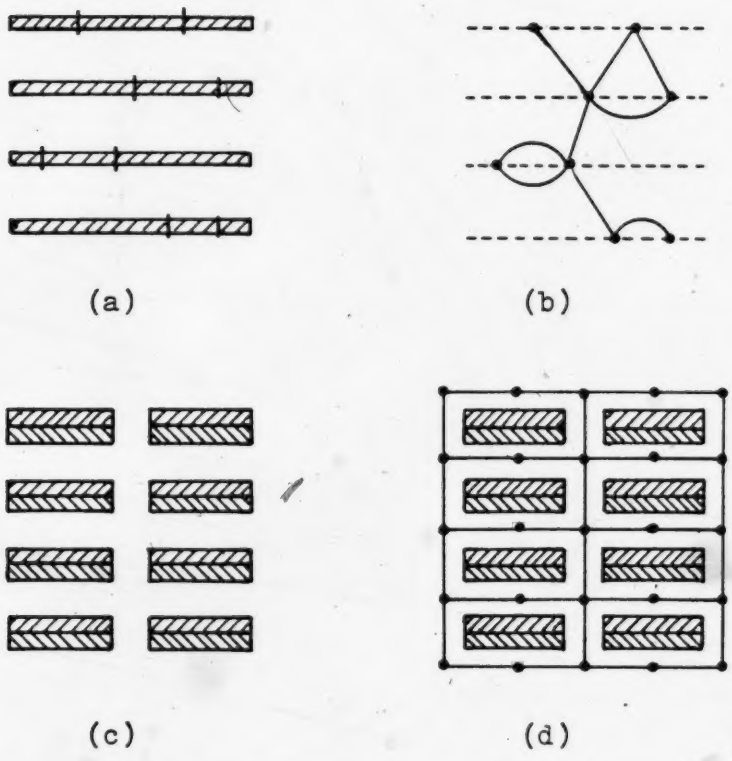


Fig. 2.6 Tree-Type Approach to Standard-Cell Global Routing

The nets are ordered based on the estimated densities of their edges, then an approximate Steiner tree is constructed for each net. In [Aos83], the edge weight is a function of the maximum local density in the channel interval between the two terminals. The router orders the nets and then, for each net, finds a minimum spanning tree.

The lack of standard-cell global routers reported in the literature can be attributed to two reasons: (1) Early standard cells were designed with single-entry terminals, i.e. terminals which appear on one side of the cell only. With such cells, the flexibility of assigning nets to

different channels is limited and therefore little emphasis was placed on global routing. (2) In standard-cell layout, the channel widths are allowed to vary so as to accommodate 100% routing. Because a solution is always guaranteed, although it may not be the most efficient, a global router is not mandatory. Consequently, some standard cell layout systems do not employ a global router at all [Meh84]. Instead, these systems use channel routers to route the channels one at a time, routing as many connections as possible for each channel.

The major weakness of all existing tree-type global routing algorithms is that they are sequential. The quality of their solutions is dependent on the order in which the nets are processed. Many criteria have been suggested for ordering the nets. Some of the ordering criteria which have been used are: (1) by the number of terminals in the net, (2) by the estimated length of the net, (3) by the perimeter or area of the smallest rectangle bounding the terminals of the net, and (4) by the estimated densities of the regions through which the net traverses. Most routers order the nets in ascending order of these measures, e.g., smaller nets are routed before large nets. Some routers, however, use the opposite ordering. As there is no consensus as to which ordering scheme is the best. It seems desirable to eliminate the need for net ordering altogether.

Another problem with some of the existing global routers is that they only produce a rough initial solution

and rely heavily on an iterative improvement step to produce a good solution. The iterative improvement step often requires large amounts of computation time. For example, the TimberWolf Package uses the simulated annealing approach in the iterative step which is very time consuming. In [Sec84], it was reported that the TimberWolf Package uses about 12 minutes of VAX 11/780 CPU time to global route a 100-cell circuit, and 2 hours for a 1500-cell circuit.

2.4 The U. of A. Standard-Cell System

A standard-cell design system has been developed at the University of Alberta to be used as a research tool for developing and testing layout algorithms. The layout model used in our standard-cell system is shown in Fig. 2.7. Standard cells of uniform height and varying widths are arranged end-to-end in horizontal rows by the placement phase. The number of cell rows is determined by the placement algorithm, and the number of channels is one less than the number of rows.

If a pair of terminals on a net are separated by a row that does not contain a terminal of that net, then a special "feedthrough" cell is inserted into that row. The feedthrough cell is simply a wire which allows the net to pass through that row. We assume that feedthroughs which are required have been inserted by either the placement algorithm or a pre-processing step before global routing so that all nets have terminals on consecutive rows.

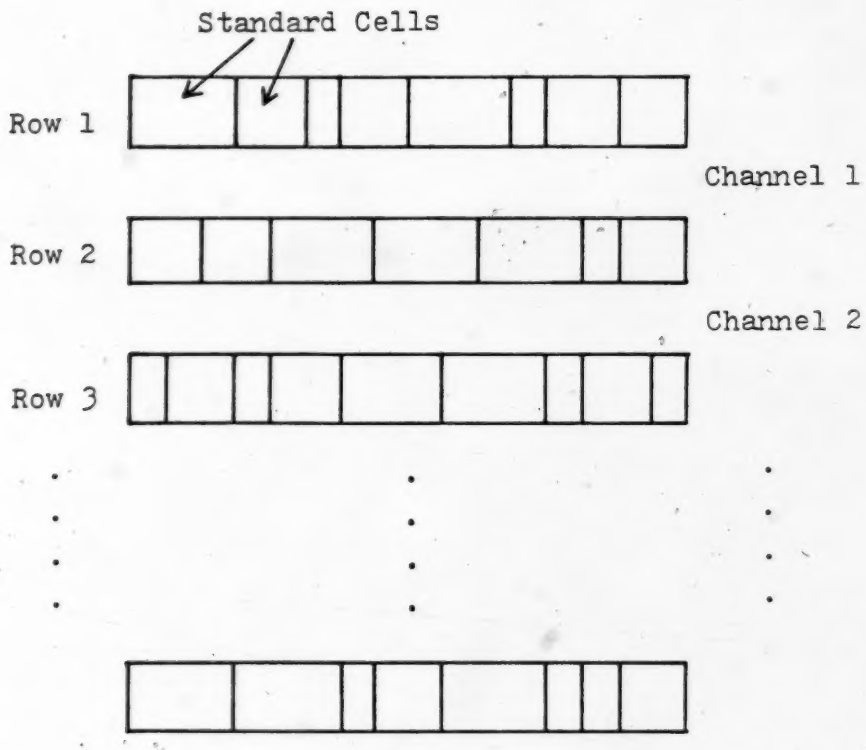
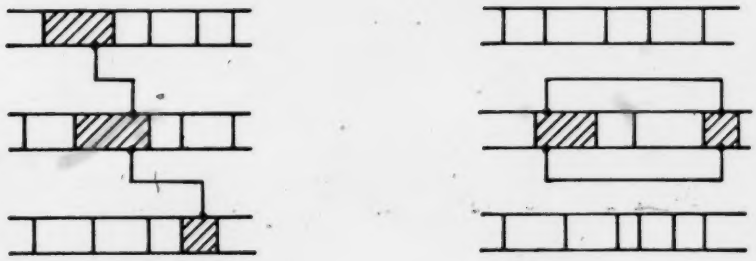


Fig. 2.7 Standard-Cell Layout Model

Double-entry cells are used in our system. With double-entry cells, the need for feedthroughs (which increase the area of a circuit) is minimized because a net can pass through any row that contains a terminal on that net (Fig. 2.8a). Another advantage of using double-entry cells is that terminals on the same row can be connected by a wire running in either the channel above or the channel below the row (Fig. 2.8b), thus increasing the ability of the global router to reduce the routing area.

The routing in our system is performed by a global router and a detailed router. The routers use the two-layer



(a) (b)

Fig. 2.8 Advantages of Double-Entry Cells

Manhattan wiring model. Only the channels are used for routing, no routing is permitted over the cells or around the end of the cell rows. We will not consider the connections to the I/O pads, therefore, the area above the top row, below the bottom row, left of the rows and right of the rows are not used for routing. We also assume that the routing of power and ground have been taken care of because the standard cells have internal horizontal power and ground buses which abut when the cells are placed in rows.

We have decided to use a gridless channel router in our system because: (1) we believe the more efficient use of silicon area allowed by a gridless channel is worth the cost of increased router complexity, and (2) a gridless router can also function as a gridded router in the special case that the terminals do lie on a uniform grid. The gridless channel router we use is based on an existing gridded channel router called the Greedy Channel Router [Riv82a]. In the next chapter, we will present the Greedy Channel Router

and describe how we have modified it to be gridless.

In view of the lack of standard-cell global routers and the drawbacks of the existing tree-type global routing approach (see Section 2.3.2), we developed a new standard-cell global router using an approach different from the tree-type approach. In Chapter 4, we will describe our new algorithm in detail.

Chapter 3

Detailed Routing

In the previous chapter, we introduced the basic concepts pertaining to channel routing. We also pointed out that the gridless approach allows more compact cells to be designed. Hence we have chosen to use a gridless channel router in our standard-cell system. This gridless router is based on an existing gridded router called the Greedy Channel Router due to Rivest and Fiduccia [Riv82a]. In this chapter, we begin by introducing the original Greedy Channel Router. Then we explain what the complications are in the gridless routing problem, and describe how we have modified the Greedy Channel Router to be gridless.

3.1 Original Greedy Channel Router

3.1.1 Problem Definition

A gridded channel routing problem is specified by: (1) channel length λ, (2) top connection list T, (3) bottom connection list B, (4) left connection set L, and (5) right connection set R. The channel length λ specifies the number of columns, located at x-coordinates $1,2,..,\lambda$, in the channel. The top connection list $T=(T_1,T_2,...,T_\lambda)$ and bottom connection list $B=(B_1,B_2,...,B_\lambda)$ specify the nets to which the terminals on the top and bottom channel edges belong.

T_i (B_i) denotes the net for the top (bottom) terminal at the i-th column from the left. If there is not a top (bottom) terminal at the i-th column, then T_i (B_i) is zero. The left connection set L and right connection set R specify the nets which must be connected to the left and right edges of the channel.

The gridded channel routing problem is, given the channel specification as described above, find for each net, a set of connected horizontal and vertical wires which form a routing path for the net such that the number of tracks used, w, to route the channel is minimum. The endpoints of the wires are grid points (x,y) with $0 \le y \le w+1$, and $0 \le x \le \lambda+1$ except when columns to the right of the channel are used. Wires from the same net which cross each other on different layers at a grid point are connected by a contact at that point.

3.1.2 The Greedy Channel Router

The major steps of the Greedy Channel Router are given in Fig. 3.1. In the following paragraphs, we will give a brief overview of the router.

The Greedy Channel Router scans the channel to be routed from left to right, column by column, completing the wiring up to the current column before proceeding to the next column. In each column, it trys to optimize the wiring produced by using greedy heuristics. It always succeeds in completing the routing because it is allowed to widen the

```
Algorithm - Greedy Channel Router
begin
(1) assign tracks to nets at the left edge of the channel;
(2) for each column i, i=1,...λ, until i>λ and there are
    no split nets do
    begin
(3)   bring the top and/or bottom nets into the channel
      with vertical wires;
(4)   free up as many tracks as possible by collapsing
      split nets with jogs;
(5)   add jogs to reduce the range of split nets;
(6)   add jogs to raise rising nets and lower falling nets;
(7)   widen channel if necessary to bring in nets which
      could not be brought in earlier;
(8)   extend horizontal wires to the next column;
    end;
end;
```

Fig. 3.1 Greedy Channel Routing Algorithm

channel as required, and it is allowed to use columns to the right of the channel.

The input for the Greedy Channel Router consists of: (1) a specification of the channel routing problem, and (2) three non-negative integer parameters: *initial-channel-width*, *minimum-jog-length*, *steady-net-constant*.

The Greedy Router begins routing the channel with the number of tracks equals to *initial-channel-width*. Whenever

there is not enough space, the router adds a new track to the channel automatically. The router makes no jogs shorter than the *minimum-jog-length*. A *jog* is defined to be a vertical wire segment which moves the signal path of a net from one track to another (Fig. 3.2). Increasing minimum-jog-length will reduce the number of jogs and hence the number of contacts but increase the number of tracks. At each column, the router classifies each net which has a terminal to the right as either *rising*, *falling*, or *steady*. A *rising* net is a net for which the nearest terminal to the right of the current column is on the top edge of the channel at column *k* and there is no terminal on the bottom edge of the channel before column *k+steady-net-constant*. *Falling* nets are defined analogously. *Steady* nets are the remaining nets. A typical selection of values for these parameters are: initial-channel-width = channel density, minimum-jog-length = channel density/4, and steady-net-constant = channel length/10.

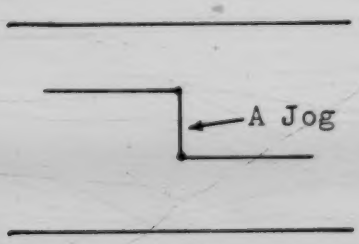


Fig. 3.2 A Jog

The main data structure of the router is a set $Y(N)$ for each net N. $Y(N)$ is the set of tracks currently occupied by net N, with each track denoted by its y-coordinate. If $Y(N)=\phi$, then the net N is not being routed currently. A net N is said to be *split* if $|Y(N)|>1$, i.e., it is occupying more than one track in the current column.

Each step of the algorithm is described in detail in the following sections.

Assign tracks to nets at the left edge of the channel

For each net N in the left connection set L, assign it to a track. Rising nets are placed above steady nets, which are placed above falling nets. The nets are placed at the center of the channel. [Riv82a] does not specify how nets of the same type should be placed relative to each other. We use the following heuristic. For rising nets, those whose first terminals are farther left are put below nets whose first terminals are farther to the the right in the channel. For falling nets, those whose terminals are farther right are placed below nets whose first terminals are farther left (Fig. 3.3). This tends to leave more free tracks in the center of the channel.

Bring the top and/or bottom nets into the channel with vertical wires

If only one of T_i or B_i is nonzero, then bring the net, T_i or B_i, into the channel if possible to the nearest possible track which is either empty or already assigned to that net (Fig. 3.4a). This is done by running a vertical

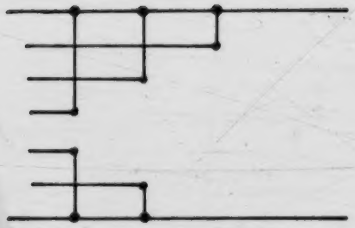


Fig. 3.3 Assign Tracks To Nets At Left Edge

wire from the terminal at the edge of the channel to the desired track. If both T_i and B_i are nonzero, then try to bring in both nets. If $T_i \neq B_i$ and the vertical wires would conflict (i.e., they overlap), then just bring in the net which can be brought in with a shorter wire (Fig. 3.4b). If there is no empty track, net $T_i = B_i \neq 0$, and there is no terminal to the right, then run a vertical wire from top to bottom (Fig. 3.4c). If $T_i = B_i \neq 0$ and there is a terminal to the right, then the net cannot be brought in. Any net(s)

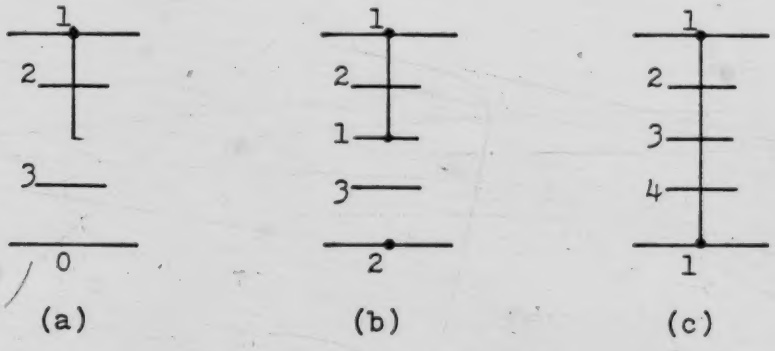


Fig. 3.4 Bring In Top and Bottom Nets

which cannot be brought into the channel in this step either because the channel is full or because of conflicts will be brought in later in line 7.

Free up as many tracks as possible by collapsing split nets with jogs

This step trys to add collapsing jogs to collapse split nets in a pattern which will create the most empty tracks in the next column. A *collapsing jog* is defined to be a piece of vertical wire which connects two tracks holding the same net without crossing another track that holds that net (Fig. 3.5a). Each split net N has $|Y(N)|-1$ collapsing jogs. A *pattern* is a set of collapsing jogs for which jogs of different nets do not overlap and there is no overlapping vertical wires placed in line 3. [Riv82a] uses an exhaustive search to find the pattern which creates the most empty tracks. We use a simple greedy approach instead of exhaustive search. For each possible collapsing jog we generate a list of collapsing jogs which are compatible to it. Two jogs are *compatible* if they do not overlap or touch at the same track (Fig. 3.5b). Each collapsing jog is assigned a weight which is the number of compatible jogs it has. This number is chosen as the weight because each collapsing jog will free up one track if the net has terminal(s) to the right, or two tracks if the net has no terminal to the right. Hence a jog with a large number of compatible jogs is likely to allow more collapsing jogs to be added, and thus is likely to free up more tracks. The

(a) Collapsing Jog (b) Compatible Jogs

Fig. 3.5 Collapsing Jogs

algorithm used in choosing the collapsing jogs is given in Fig. 3.6.

Add jogs to reduce the ranges of split nets

For each split net N with $|Y(N)|>1$, the router finds the highest and the lowest tracks in $Y(N)$. The router then trys to reduce the range of the split net by adding jogs to move the signal path of the net from the highest track to the lowest possible empty tack, and from the lowest track to the highest possible empty track (Fig. 3.7). Only jogs which are longer than minimum-jog-length and do not overlap existing vertical wires in the column are added.

Add jogs to raise rising nets and lower falling nets

All the unsplit rising and falling nets are sorted in decreasing order of the distances from the tracks they occupy to their target edges. The *target edge* of a rising net is the top edge; the target edge of a falling net is the bottom edge. In sorted order, the router trys to add vertical jogs to move the signal path of an unsplit net to

Algorithm - Select Collapsing Jogs

begin

(1) $J_1 \leftarrow$ set of possible collapsing jogs which do not overlap existing vertical wires;

(2) **while** $J_1 \neq \phi$ **do**

begin

(3) for each jog in J_1 find its compatible jogs and determine its weight;

(4) select the jog k from J_1 with the largest weight;

(5) add jog k to the column;

(6) $J_2 \leftarrow$ set of jogs compatible with jog k;

(7) $J_1 \leftarrow J_1 \cap J_2$;

end;

end;

Fig. 3.6 Choose Collapsing Jogs Algorithm

(a) Before (b) After

Fig. 3.7 Reduce Range of Split Nets

an empty track which is as close to its target edge as possible (Fig. 3.8). Only jogs which are longer than the

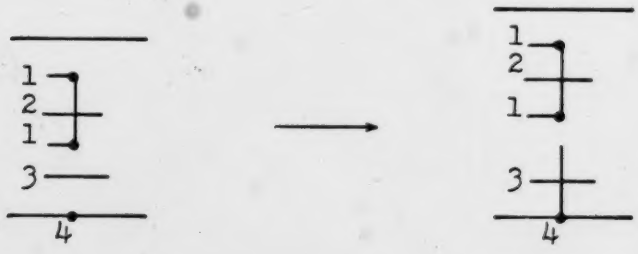
(a) Before (b) After

Fig. 3.8 Raise or Lower Nets

minimum-jog-length and do not overlap existing vertical wires are added.

Widen Channel if necessary

If a net, T_i or B_i, was not brought into a track in line 3 of Fig. 3.1, then a new track is created for that net. The router places this track as close to the center of the channel as possible provided that a vertical wire can be brought in to that track from the edge (Fig. 3.9). If a new track is inserted between track k and $k+1$, then the



(a) Before (b) After

Fig. 3.9 Widen Channel If Necessary

y-coordinates of all old tracks above k are incremented by one.

Extend to the Next column

For each unsplit net N which has no terminal to the right of the current column, $Y(N)$ is set to be the empty set. All other "dangling ends" of the nets are extended to the next column with horizontal wire segments (Fig. 3.10).

3.1.3 Advantages and Disadvantages of the Greedy Channel Routing Algorithm

The Greedy Channel Routing Algorithm has the following advantages over other existing channel routers. First, it can handle channel routing problems with cyclic constraints. Most of the other channel routers cannot route channels with cyclic constraints or need non-trivial modifications in order to be able to handle them. Secondly, it is simple, intuitive, and its control structure is flexible. Therefore, it is easy to modify or to add new rules to handle cases

(a) Before (b) After

Fig. 3.10 Extend To Next Column

like switchbox routing [Luk83], obstacle avoidance [Ham84], and gridless routing. However, the Greedy Channel Router tends to use more contacts than other routers. This is because, firstly, it allows jogs at every column. As shown in Fig. 3.11a, every jog introduces two contacts. Secondly, the Greedy Channel Router is "greedy", and hence it only focuses on the local problem at the current column without considering the global situation, thus it may introduce too many jogs and hence too many contacts (Fig. 3.11b). Another problem with the Greedy Channel Router is that, like many existing channel routers, it is gridded and hence cannot handle gridless routing problems.

3.2 Gridless Greedy Channel Router

3.2.1 Problem Definition

A gridless channel routing problem is specified by: (1) channel length λ, (2) top terminal list T, (3) bottom

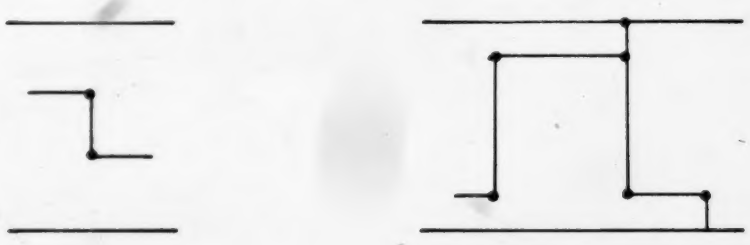


Fig. 3.11 Disadvantage of Greedy Channel Router

terminal list B, (4) left connection set L, (5) right connection set R, and (6) minimum inter-wire spacing τ. The channel length λ specifies the real horizontal length of the channel. Real horizontal coordinates start from x=0 at the left edge of the channel to x=λ at the right edge. Terminals may lie along the top and bottom channel edges at any abscissa $0 \le x \le \lambda$. Top terminal list $T=(t_1,t_2,\ldots,t_m)$ where m is the number of terminals on the top channel edge, and t_i, $1 \le i \le m$, denotes the i-th top terminal from the left. Bottom terminal list $B=(b_1,b_2,\ldots,b_n)$ where n is the number of terminals on the bottom channel edge, and b_i, $1 \le i \le n$, denotes the i-th bottom terminal from the left. Each terminal, t_i or b_i, is described by a 2-tuple (x,N) where x is the x-coordinate of the terminal, and N is the net to which the terminal belongs. The left connection set L and right connection set R are the same as in the gridded channel routing problem. τ is the minimum allowable separation between wires. It should be at least the contact-to-contact separation specified by the design rules. We will assume that the terminals in the lists T and B are sorted according to their x-coordinates, and no two terminals on the same side of the channel are closer than τ units apart.

The gridless channel routing problem is, given the channel specification as described above, find a routing path for each net so that the minimum amount of space is used to route all the nets. All horizontal wires used in the routing must lie on tracks (horizontal grid lines) whereas

the vertical wires may run along any abscissa x, provided that there are no design rule violations and the routing paths for all nets are disjoint.

3.2.2 Extension of the Greedy Channel Router to a Gridless Channel Router

The main problem with gridless routing arises when terminals on opposite sides of a channel are too close together to allow vertical wires to be routed from both terminals, as shown in Fig. 3.12a and Fig. 3.12b. Both routings in Fig. 3.12a and Fig. 3.12b violate design rules. To overcome these problems, we redefine the notion of columns, relax our wiring model, and add new rules to the Greedy Channel Routing algorithm to make it into a gridless router.

First, we consider how the columns for a gridless channel should be defined. If there is a terminal on both sides of a gridded channel at a column, the Greedy Channel Router would bring the top and bottom net into the channel with vertical wires which do not overlap as shown in Fig. 3.12c. Notice that the problem in the gridless channel of Fig. 3.12a can be avoided by this same mechanism if we form a column with AB and a column with BC. This requires that we redefine the notion of columns as follows: (1) Every terminal is on one or two columns. (2) Two terminals on opposite sides of a channel which are closer than τ are on the same column. (3) As many columns as possible are added

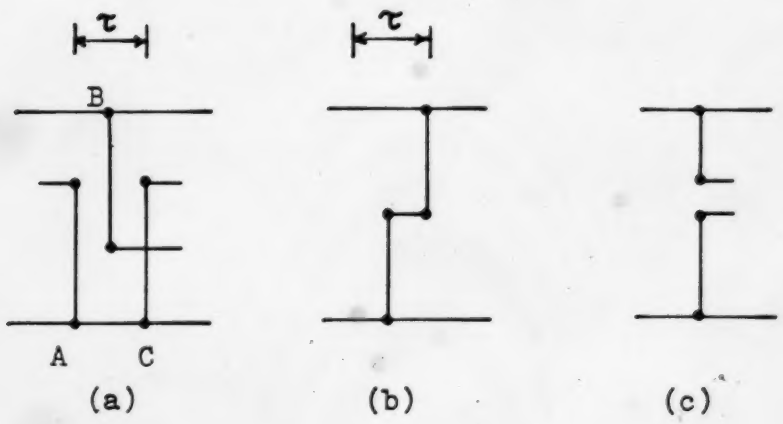


Fig. 3.12 Problems of Gridless Routing

between terminals such that these columns are τ units apart.

Since the Greedy Channel Router operates in a column-by-column manner, and the terminals in lists T and B for the gridless channel are not organized in terms of columns, we need an algorithm to generate the columns from T and B.

The column generation algorithm is given in Fig. 3.13. The input to the column generation algorithm consists of: (1) top terminal list T, (2) bottom terminal list B, and (3) the minimum inter-wire spacing τ. The task of the algorithm is to generate a list of columns that satisfy our new notions of columns. We will briefly describe how this algorithm generates the columns.

The algorithm first merges T and B into a merged terminal list $P=\{p_1,p_2,\ldots,p_{m+n},0,0\}$ where p_i denotes the i-th terminal in the list, and 0 denotes a null terminal. Two null terminals are appended to P to facilitate the

```
Algorithm - Column Generation
begin
(1) merge top connection list T and bottom connection list B
    to form merged list P={p1,p2,....pm+n,0,0};
(2) i ← 0; STOP ← false;
(3) while STOP≠true do
    begin
(4)   i ← i+1;
(5)   if pi≠0 and pi-1=0 and pi is not covered then
      begin
(6)     form column |'i or |i;
(7)     STOP ← true;
      end;
(8)   if pi=0 and pi+1=0 then STOP ← true;
(9)   d ← x(pi+1) - x(pi);
(10)  if d=0 then form column |'i+1;
(11)  if 0<d<τ then form column /'i or \'i+1;
(12)  if τ≤d<2τ and pi is not covered then form |'i or |i;
(13)  if d≥2τ then
      begin
(14)    if pi is not covered then form column |'i or |i;
(15)    insert ⌊d/τ⌋-1 empty columns, |, between pi and
        pi+1, τ units apart, starting at x(pi)+τ;
      end;
    end;
end;
```

Fig. 3.13 Column Generation Column

detection of the end of P. Let us also define a function x such that $x(p_i)$=x-position of p_i. T and B are merged so that the p_is are sorted by their x-positions, i.e., $x(p_{i-1})\leq x(p_i)\leq x(p_{i+1})$, $2\leq i\leq m+n-1$, and if $x(p_i)=x(p_{i+1})$, then p_i is a top terminal and p_{i+1} is a bottom terminal.

Throughout the algorithm, we scan P from left to right, focusing on a pair of adjacent terminals p_i and p_{i+1}, as possible candidates to form a column. We denote a column with the symbols |, /, and \. | denotes a vertical column, / and \ denote oblique columns. A vertical column may be of four forms: $|_j^k$ has top terminal p_k and bottom terminal p_j (Fig. 3.14a), $|^j$ has only a top terminal p_j (Fig. 3.14b), $|_j$ has only a bottom terminal p_j (Fig. 3.14c), and | is empty (Fig. 3.14d). An oblique column, $/_j^k$ or $\backslash_k^j$, always has two terminals (Figs. 3.14e, f). A terminal is said to be *covered* if it is in a column.

In the algorithm, we use the flag *STOP* to indicate when we have reached the end of P. Line 2 initializes i and *STOP*.

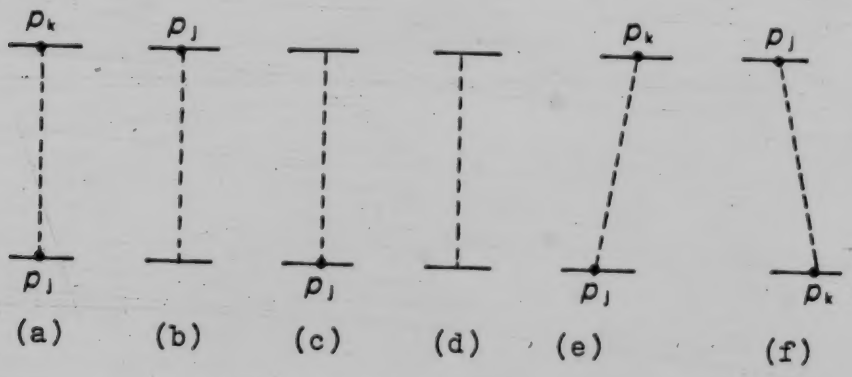


Fig. 3.14 Gridless Columns

The loop between line 3 and line 15 scans the list P. Lines 5 to 8 detect the end of P. If we have not reached the end of P yet, then we calculate d, the x-separation between p_i and p_{i+1}. If $d=0$, i.e., p_i and p_{i+1} are vertically aligned, then we form a vertical column $|^{i+1}_{i}$ (Fig. 3.15a). If $0<d<\tau$, i.e., p_i and p_{i+1} are not vertically aligned but are closer than τ units apart, then we put them in the same oblique column, $/^{i+1}_{i}$ or $\backslash^{i+1}_{i}$ (Figs. 3.15b, c). If $\tau \le d < 2\tau$, and p_i has not been covered yet, then we put p_i in column $|_i$ or $|'$ (Figs. 3.15d, e). If $d \ge 2\tau$, then there is room to insert $\lfloor d/\tau \rfloor - 1$ empty columns, $|$, between p_i and p_{i+1}. These empty columns are placed, starting at $x(p_i)+\tau$, τ units apart each (Fig. 3.15f). Before these columns are inserted, p_i is put in a column if it has not been covered.

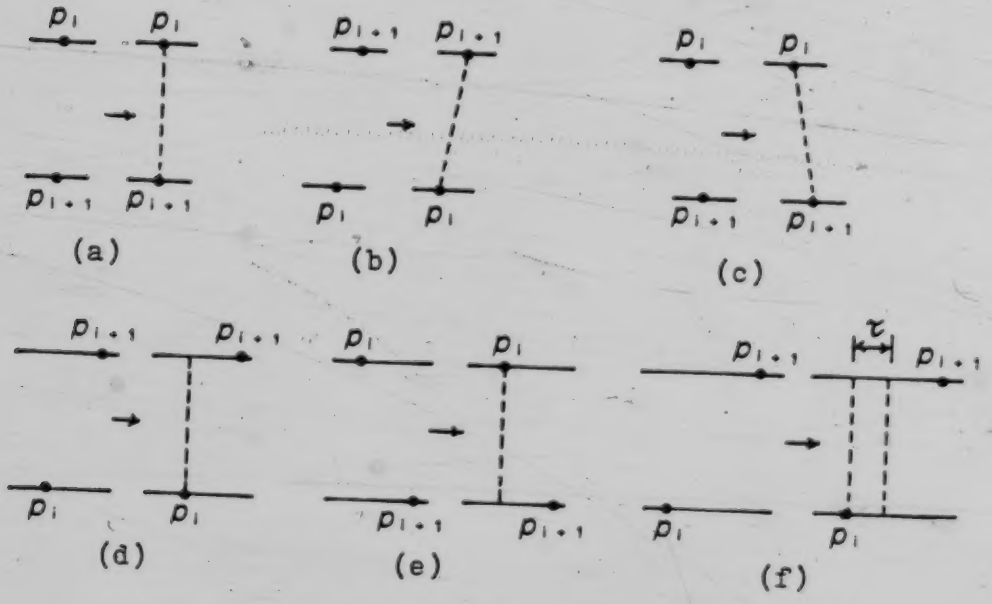


Fig. 3.15 Column Generation

Fig. 3.16 shows an example of the columns generated by the column generation algorithm. Note that two neighboring columns may share a terminal. In the gridded case, vertical wires in different columns do not interfere with each other because they are always at least τ units apart. In the gridless case, the vertical wiring in two columns which share a terminal may interfere with each other, therefore special attention has to be paid to columns which share a terminal. Since the Greedy Channel Router scans a channel from left to right, the wiring at a column for the gridless case can be done the same as for the gridded case if the column does not share a terminal with the previous column. If a column shares a terminal with the previous column, then the router must remember what vertical wires were put in the previous column and should not use wires that will create conflicts in the current column. For example in Fig. 3.17a, column *AB* is independent of the previous column, therefore it can be wired independently. However, column *BC* shares

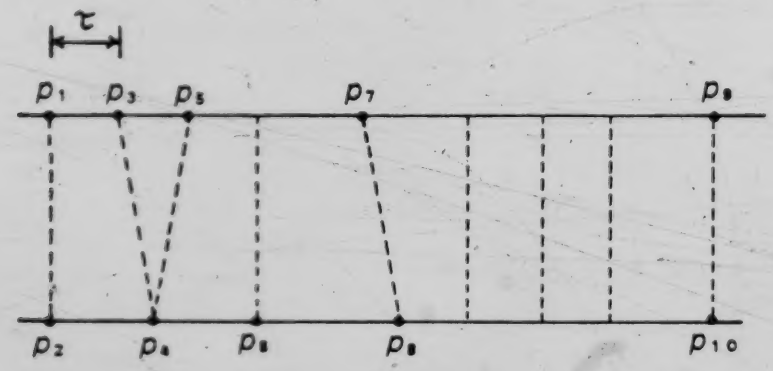


Fig. 3.16 Example of Column Generation

terminal *B* with column *AB*, hence the router must not put any vertical wires in column *BC* which will interfere with the vertical wires in column *AB*. Fig. 3.17b shows a legal wiring for the situation in Fig. 3.17a.

Besides redefining the notion of columns, and paying special attention to adjacent columns which share a terminal, the two-layer Manhattan wiring model we use must also be relaxed in order to handle the gridless case. In a gridded channel, two terminals opposite each other in a column belonging to the same net can be connected by a vertical wire running straight across the channel as shown in Fig. 3.18a. In the gridless case, however, the two terminals may not be at the same x-position, as shown in Figs. 3.14e, and 3.14f. They cannot be connected by a straight wire because only horizontal and vertical wires are allowed. On the other hand, connecting them as in Fig. 3.12b violates design rules because the two contacts would be too close together. In order to join the two vertical wire

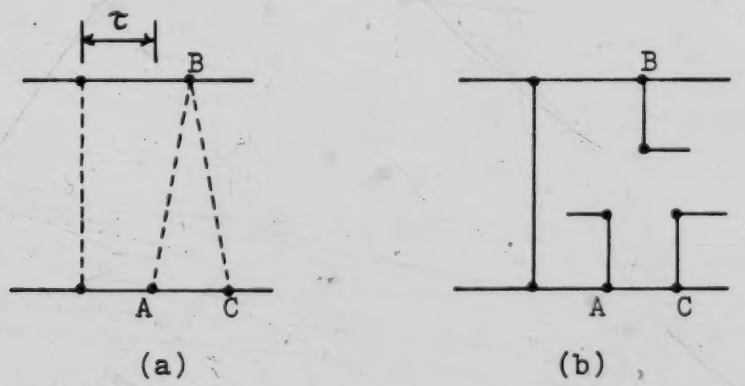


Fig. 3.17 Neighboring Columns Which Share a Terminal

segments which bring the nets into the channel, we relax our wiring model by allowing short horizontal runs of polysilicon. These horizontal polysilicon wires run along horizontal tracks and metal wires are allowed to run over them on the same track. This relaxation is necessary if only horizontal and vertical wires are allowed. The wiring for the situation shown in Fig. 3.12b using this relaxed model is shown in Figs. 3.18b, and 3.18c.

3.2.3 The Gridless Greedy Channel Router

With the extensions described above, we have modified the Greedy Channel Router so that it can handle gridless channel routing problems. In our gridless channel routing algorithm, the notions of channel width, minimum-jog-length, steady-net-constant, etc., remain unchanged. The major steps of our gridless channel router are given in Fig. 3.19. Note that the control structure is almost the same as the original Greedy Channel Router in Fig. 3.1, but new rules

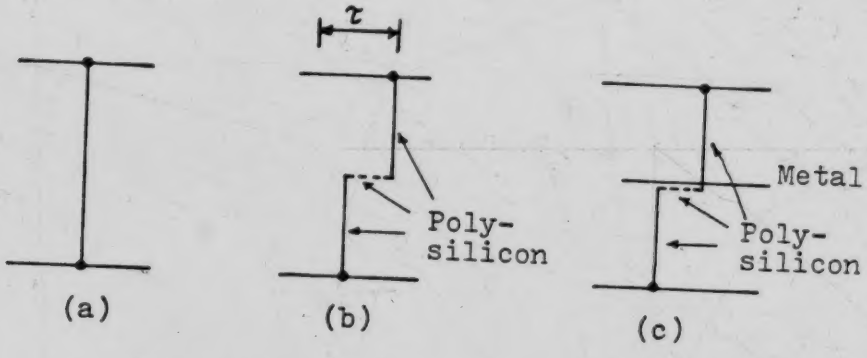


Fig. 3.18 Relaxed Wiring Model

```
Algorithm - Gridless Channel Router
begin
(1) generate columns; let C=number of columns generated;
(2) assign tracks to nets at the left edge of the channel;
(3) for each column i, i=1,...,C, until i>C and there are
    no split nets do
    begin
(4)   bring the top and/or bottom nets into the channel
      with vertical wires;
(5)   free up as many tracks as possible by collapsing
      split nets with jogs;
(6)   add jogs to reduce the range of split nets;
(7)   add jogs to raise rising nets and lower falling nets;
(8)   widen channel if necessary to bring in nets which
      could not be brought in earlier;
(9)   extend horizontal wires to the next column;
    end;
end;
```

Fig. 3.19 Gridless Channel Routing Algorithm

have been embedded into the various steps of the algorithm.

In our gridless routing algorithm, we use the column generation algorithm to generate the columns first before applying the routing heuristics. The step of assigning tracks to nets at the left edge is not changed from the gridded case. Additional rules are added to the step where we make connections to the top and bottom terminals.

Vertical wire segments which bring in nets from the channel edges are placed at the x-position of the terminals. If the current column shares a terminal with the previous column, then the router remembers the vertical wires in the previous column. Vertical wires will be placed in the current column such that they do not overlap or come too close to wires in the previous column. If the situation of Fig. 3.12b occurs, horizontal polysilicon wires are used as in Figs. 3.18b and 3.18c. The step of collapsing split nets is also modified. It may happen that the jog to be added is not vertical as illustrated in Figs. 3.20a and 3.20c. In that case, the jog is implemented in the fashion shown in Figs. 3.20b and 3.20d. In the next two steps, reducing the range of split nets and raising/lowering nets, jogs which are used do not conflict with wires in the the previous column. In the step which widens the channel, the new rules in line 4 are observed when bringing in the nets with vertical wires. When extending to the next column, if the current column shares a

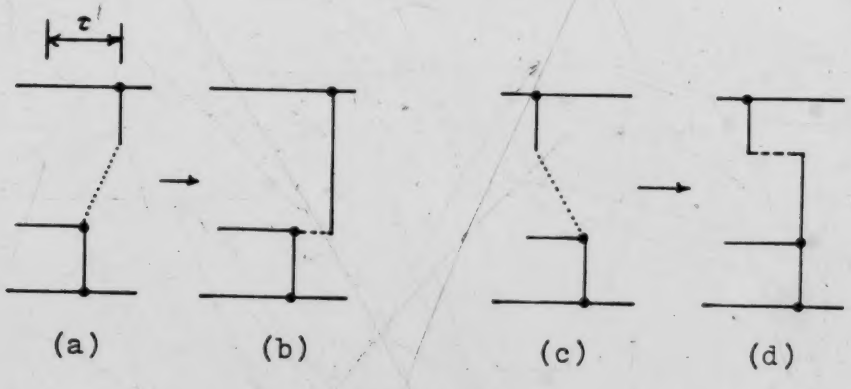


Fig. 3.20 Gridless Collapsing Jogs

terminal with the next column, then the router will remember the vertical wires placed in this column.

Fig. 3.21 shows a gridless channel routed by our gridless channel router. In the figure, we circled the places where the complications of gridless routing arise. With the new rules that we introduced, our router is able to route the channel. It should be pointed out that our gridless channel router can also function as a gridded router if the input problem is gridded. The solution produced would be identical to that of the Greedy Channel Router.

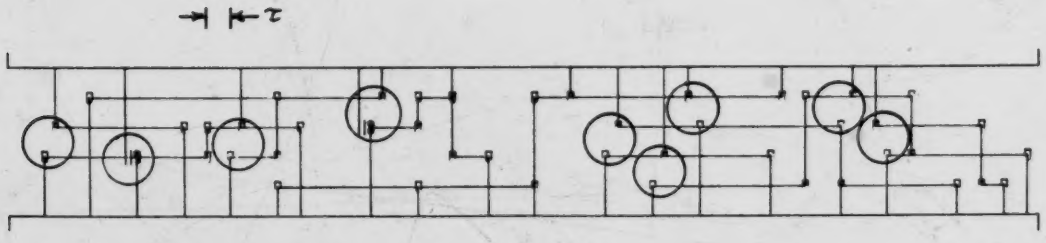


Fig. 3.21 Example of Routed Gridless Channel

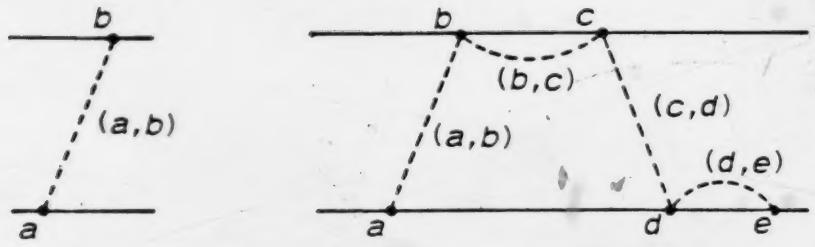
Chapter 4

Global Routing

As discussed in Chapter 2, almost all existing global routing algorithms use the sequential tree-type approach. We have developed a standard-cell global routing algorithm which uses a new approach. This chapter presents this algorithm which we will call the track-filling algorithm. We begin by defining the terms that will be used in the formulation of the problem and in the discussion of our algorithm. Then we present the details of our algorithm.

4.1 Definitions and Concepts

Let us consider a net N from a placed circuit which is to be routed. From the placement of cells and the net-list of the circuit, we can determine the locations of the terminals of N. Let x be a function such that $x(a)$=x-position of terminal a. In each channel that contains at least two terminals of N, we order the terminals of N by their x-positions. We will decompose N into a set of net segments in each channel. We define a *net segment* of net N in channel C to be a pair of terminals on N in C. If a and b are two terminals on N in C, $x(a) \leq x(b)$, then we denote the net segment by (a,b) (Fig. 4.1a). The *canonical set of net segments* which covers net N in channel C is the set of all pairs of adjacent terminals on N in C (Fig. 4.1b).



(a) A Net Segment (b) Canonical Net Segments

Fig. 4.1 Net Segments of A Net

A net segment can be implemented with one horizontal wire and two vertical wires in detailed routing (Fig. 4.2). Since the x-positions of the terminals are fixed, and because in global routing we are not concerned with the exact positions of the wires, it suffices to consider only the horizontal wire which implements a net segment. We say that a net segment is implemented as a wire segment where a *wire segment* of a net N in channel C is a piece of horizontal wire which runs between a pair of terminals on N in C. If a and b are two terminals on N in C, $x(a) \leq x(b)$, then we denote the wire segment by $w(a,b)$ (Fig. 4.3a) and say $w(a,b)$ *connects* terminals a and b. The *span* of a wire segment $w(a,b)$ is the interval of the channel lying between the terminals a and b of the segment. The *segment density* of a wire segment $w(a,b)$ in a channel is the maximum local density along the span of the segment between a and b. The *canonical set of wire segments* which covers N in C is the

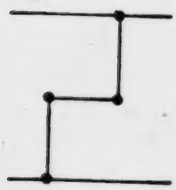


Fig. 4.2 Implmentation of a Net Segment

set of wire segments which implement the canonical set of net segments covering net N in channel C (Fig. 4.3b).

Note that all terminals on N in C are part of at least one wire segment and no wire segments in the canonical set overlap. Also note the distinction between a net segment (a,b) and a wire segment $w(a,b)$. The net segment refers to the pair of terminals a, b only, whereas the wire segment refers to a horizontal wire between a and b.

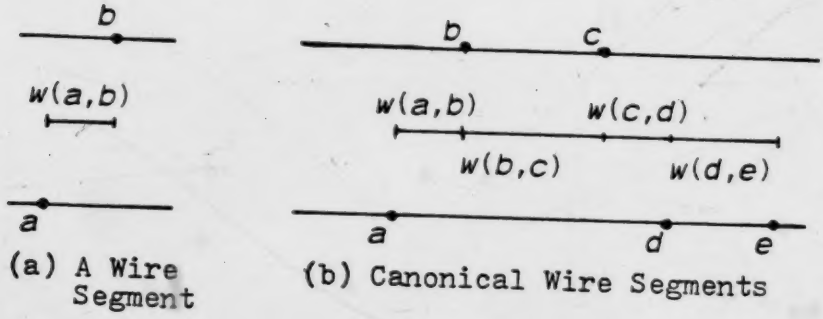


Fig. 4.3 Wire Segments of a Net

We broadly classify the canonical wire segments into two major types: *cross-channel* segments and *same-row* segments. A *cross-channel* segment is a segment which connects two terminals that are on different rows (Fig. 4.4a). A *same-row* segment is one which connects two terminals on the same row (Fig. 4.4b).

We may further classify the cross-channel segments into two types: *essential* and *non-essential cross-channel* segments. If a cross-channel segment in a channel is the only cross-channel segment for that net in that channel, then we say it is an *essential cross-channel* segment (Fig. 4.4c). A cross-channel segment which is accompanied by some other cross-channel segments for the same net in the same channel is not essential, we say it is a *non-essential cross-channel* segment (Fig. 4.4d).

Same-row segments can also be classified into two types: *switchable* and *non-switchable same-row* segments. If a pair of terminals at either end of a same-row segment in a channel can be connected by another same-row segment in an adjacent channel, then the two same-row segments are called *switchable same-row* segments; we say these two segments are *partners* of each other (Fig. 4.4e). Very often, two terminals on the same row connected by a same-row segment in a channel are connected by more than one segment in the adjacent channel as shown in Fig. 4.4f. We say the same-row segment is *non-switchable*. Another situation where non-switchable same row segments arise is when the two

terminals are either on the top row or on the bottom row. Since the area above the top row and below the bottom row are not used for routing, these same-row segments are non-switchable (Fig. 4.4g). We will refer to a non-switchable segment whose terminals are on the top row as a *top non-switchable* segment, one whose terminals are on the bottom row as a *bottom non-switchable segment*.

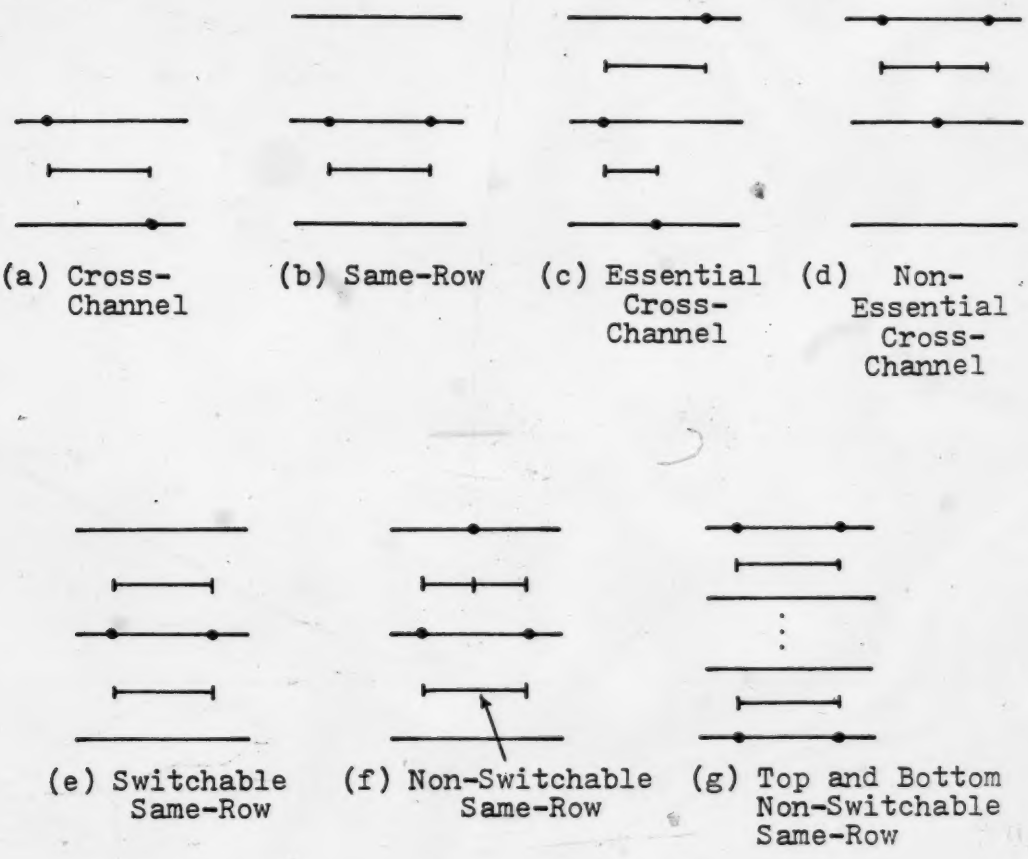


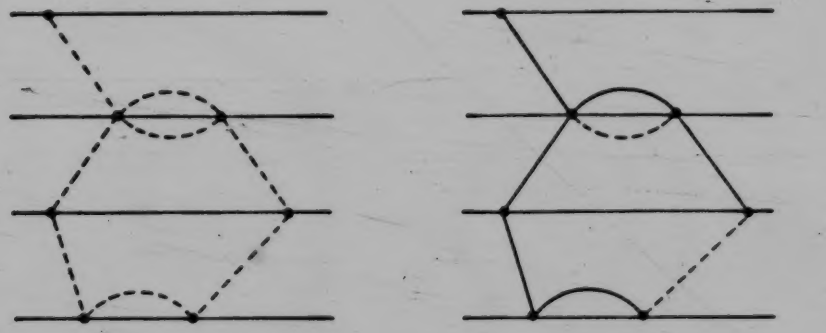
Fig. 4.4 Classification of Wire Segments

4.2 Problem Definition

Having defined the concepts of net segments, wire segments and the canonical coverings, we may now consider the problem of global routing in terms of these concepts.

Consider a graph corresponding to a net N in which each vertex is a terminal of N and each edge is a canonical wire segment connecting two terminals (Fig. 4.5a). Since a graph with n vertices can always be connected by $n-1$ edges which form a spanning tree covering all vertices of the graph, a n-terminal net can always be connected by $n-1$ wire segments (Fig. 4.5b). The problem of global routing is thus equivalent to finding such a tree for every net N using canonical wire segments of N.

We formulate the standard cell global routing problem as follows:



(a) Graph represents a net

(b) A tree spanning the net's terminals

Fig. 4.5 Graph Representation of Net Segments

Given: a cell placement and net-list of a circuit

Problem:

Let S be the set which contains the union of all canonical sets of wire segments of net N in the circuit. For each net N with n terminals and whose S contains m ($m \geq n-1$) wire segments, select from S $n-1$ segments which form a spanning tree covering N's terminals. Let D_i denote the density of channel i, $i=1,\ldots,NC$, NC=number of channels, after all nets have been connected. Our objective is to minimize the total channel density, ΣD_i.

Before proceeding to the description of our global routing algorithm which attempts to solve the above problem, we shall first present a simple example to illustrate the importance of global routing. From this example, the reader may obtain a basic understanding of what global routing is about, and why a good global routing is preferable to a bad one.

Consider Fig. 4.6a. Terminals 1 and 2 belong to net 1. Terminals 3, 4, 5, 7, 9 belong to net 2. Terminals 6, 8, 10 belong to net 3. The net segments of each net are drawn in dotted lines. Net 1 has only one segment: (1,2) in channel 1. Net 2 has six segments: (4,3), (3,5) in channel 1, (4,7), (7,9), (9,5) in channel 2, (7,9) in channel 3. Net 3 has four segments: (6,8), (8,10) in channel 2, (6,8), 8,10) in channel 3. Net 1 can be routed in only one way. Both nets 2 and 3 can be routed in a number of ways. Fig. 4.6b shows a global routing in which each channel has density 2.

Fig. 4.6c shows another global routing in which channel 1 has density 1, both channels 2 and 3 have density 2. In Fig. 4.6d, all channels have density 1. It is obvious that Fig. 4.6d is better than Figs. 4.6b and 4.6c because it has the least total channel densities.

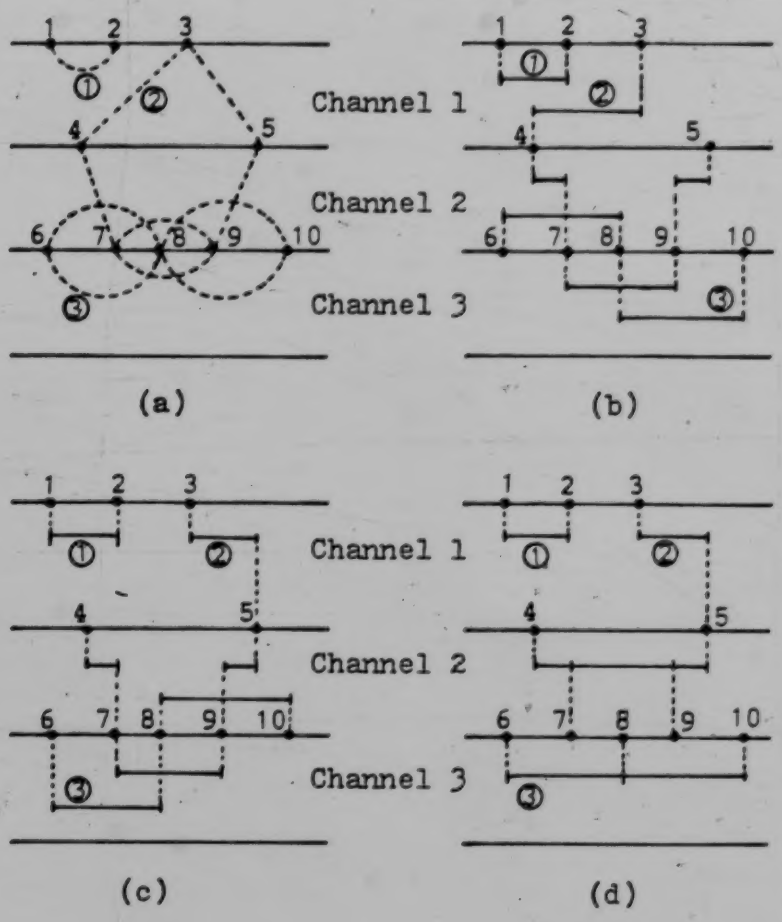


Fig. 4.6 An Global Routing Problem Example

4.3 A Track-Filling Approach to Standard-Cell Global Routing

We propose a new approach which we call the *track-filling* approach to the standard-cell global routing problem. In this approach, we define a *track* to be a set of non-overlapping wire segments from a single channel (Fig. 4.7). Note that this definition of a track is different from the definition used previously in channel routing. "Track-filling" refers to the process of building up tracks in the channels by selecting wire segments from the sets of canonical wire segments in each channel.

We have developed a new standard-cell global routing algorithm using this track-filling approach. We will refer to this algorithm as the *track-filling algorithm* hereafter. In brief, the algorithm first builds the canonical sets of wire segments in each channel, then orders the channels in some sequence. The track-filling step will cycle through the channels in the determined sequence and for each channel, builds a track of wire segments from left to right. The

Fig. 4.7 A Track used in Track-Filling Approach

algorithm builds tracks in this fashion until all nets are connected. This algorithm is a parallel routing algorithm because it does not process the nets one at a time. Hence, it avoids the disadvantage of the tree-type algorithms, i.e., the need to order the nets and process them in that order.

4.4 The Track-Filling Algorithm

The track-filling algorithm is given in Fig. 4.8. In the following sections, we will discuss the algorithm in detail.

The first step of the algorithm is to build, for each channel, the set S_1 containing the union of all the canonical wire segments for all nets in the channel. The segments in each S_1 are then sorted by the x-positions of their left endpoints. Next, each segment is classified into one of the four types defined above: (1) essential cross-channel, (2) non-essential cross-channel, (3) switchable same-row, and (4) non-switchable same-row.

After the S_1 of each channel has been built, the density and span of each channel is estimated by assuming that all segments in S_1 will be routed in the channel. Next, the channel ordering step determines a sequence in which the track-filling step will cycle through the channels. Because the track-filling step will build a track in each channel on a given cycle, we believe that the order in which the channels are cycled does not have a great impact on the

```
Algorithm - Track-Filling Global Routing Algorithm
begin
(1) build up canonical sets, S1, of wire segments for each
    channel;
(2) estimate the density and span of each channel;
(3) order the channels by decreasing order of estimated
    densities;
(4) while not all S1s are empty do
     begin
(5)   for each channel in the sorted order do
       begin
(6)     remove redundant segments from S1;
(7)     update channel density and segment density
        estimates;
(8)     while not end of track do
         begin
(9)       build mutually exclusive set of wire segments E;
(10)      rank all segments in E;
(11)      select segment w with the highest rank from E;
(12)      add w to S2;
(13)      delete w from S1;
(14)      update global and local subnets;
         end;
       end;
(15)  if no segment is selected then  FMAX ← FMAX+0.1;
     end;
end;
```

Fig. 4.8 Track-Filling Algorithm

quality of the results produced. The channels may be ordered either in descending or ascending order of their estimated densities. Currently, we use descending order in our algorithm.

4.4.1 Mutually Exclusive Set of Wire Segments

The track-filling step cycles through the channels in the order determined by the channel ordering step, and builds a track for each channel. The channels are cycled until all nets are connected. A track is built by adding segments to it from left to right. Each segment added to the track is selected from a set of mutually exclusive wire segments. The *mutually exclusive set*, E, is a subset of S_1 that consists of all wire segments from S_1 which are to the right of all the segments in the current track, and whose spans all overlap one another. Heuristics, which we will describe later, are used to select a segment from E. The segment selected from E is deleted from S_1, and added to another set, S_2, which contains the tracks in the channel. We say the segments in S_2 are *selected segments*, and the segments in S_1 are *unselected segments*. An unselected segment is said to be *redundant* if its two terminals are already connected by selected segments, i.e., there is already a signal path between the terminals using segments in S_2. When an unselected segment in S_1 becomes redundant, it is deleted from S_1. The detection of redundant segments will be described later in Section 4.4.4.

4.4.2 Fullness & Density Estimates

To facilitate the selection of a segment from the mutually exclusive set, we have introduced the notion of *fullness*. The *local fullness* of a channel at a horizontal position x is defined to be

local fullness at x = local density at x/channel density.

Similarly, the fullness of a channel interval between horizontal positions x_1 and x_2 is defined as the maximum local fullness between x_1 and x_2, i.e.,

fullness between x_1 and x_2
= maximum local density between x_1 and x_2/channel density,

and the *segment fullness* of a wire segment is defined as

segment fullness = segment density/channel density.

Since local density ≤ channel density, we have 0≤fullness≤1. We say a channel interval is full if its fullness f=1, and empty if f=0. A channel interval with fullness f_1 is said to be more full than another interval with fullness f_2 if $f_1>f_2$ and less full if $f_1<f_2$.

Normally, the maximum fullness is 1. However, we introduce the notion of maximum allowable fullness, *FMAX*, *FMAX*>0. A channel interval with fullness f>*FMAX* is considered to be full. For example, if *FMAX*=0.5, then a channel interval with fullness f=0.55 is considered full. If *FMAX*>1, then no channel interval will be considered full

because the maximum fullness is 1, which is less than *FMAX*.

Since the local and channel densities are not known until after global routing, we have to estimate them in order to measure fullness during global routing. Different density estimate methods may be used. We have used three during the development of this algorithm. In the following discussion, *df*, *Df* denote the final local density and channel density after global routing respectively. The three density estimates used are:

1. Estimate method 1: d_1, D_1

   The local densities d_1 are computed by assuming that both the unselected segments in S_1 and the selected segments in S_2 will be routed in the channel. Channel density D_1 is the maximum of d_1 over the channel. d_1 (D_1) is always higher than *df* (*Df*) because many of the unselected segments will become redundant. As the algorithm proceeds, d_1 (D_1) decreases monotonically, reaching *df* (*Df*) when global routing finishes.

2. Estimate method 2: d_2, D_2

   The local densities d_2 are computed by assuming that only the selected segments in S_2 will be routed in the channel. Channel density D_2 is the maximum of d_2 over the channel. At the start of global routing, both d_2 and D_2 are zero because S_2 is empty. As the algorithm proceeds and segments selected, d_2 (D_2) increases monotonically, reaching *df* (*Df*) when global routing finishes.

3. Estimate method 3: $d_3=(d_1+d_2)/2$, $D_3=(D_1+D_2)/2$

In this method, we use d_1 and d_2 to compensate for each other by taking their arithmetic average, i.e., $d_3=(d_1+d_2)/2$. Since $d_1 \le df \le d_2$, we expect d_3 to be a better estimate of df than both d_1 and d_2 are. Similarly, $D_3=(D_1+D_2)/2$. This estimate method is equivalent to computing the densities by considering all the selected segments and half of the unselected segments. To see why this is a reasonable property, consider Fig. 4.9 where there are p selected segments and q unselected segments passing through horizontal position x. Using the first two estimate methods, we have $d_1=p+q$ and $d_2=p$. Since global routing is not finished yet, we can expect r, $0 \le r \le q$, of the q unselected segments will be selected, making the final density $df=p+r$. Exactly how many will be selected cannot be known without actually performing the routing, therefore we assume that on average, about half of the q unselected segments will be selected, then we have expected final density$=p+q/2=(p+q+p)/2=(d_1+d_2)/2=d_3$.

4.4.3 Selection of Segments From the Mutually Exclusive Set

The segments in the mutually exclusive set are categorized into 10 classes, *CLASSes 1* to *10*, which are shown in Fig. 4.10. In Fig. 4.10, $w(a,b)$ is a segment in the mutually exclusive set, f denotes the segment fullness of $w(a,b)$ (fullness of the current channel interval between a

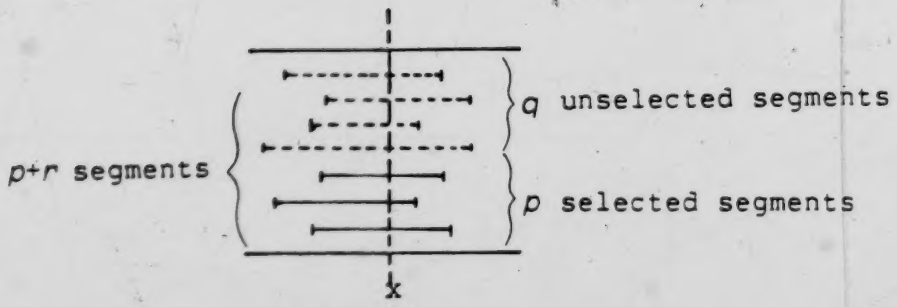


Fig. 4.9 Three Different Density Estimates

and *b*), *fa* denotes the fullness of the adjacent channel interval between *a* and *b*. Each segment is categorized according to (1) its type, (2) its fullness, and (3) the fullness in adjacent channels.

A wire segment is put into *CLASSes 1* to *9* only if it is not located in a channel interval which is full (i.e., *f<FMAX*). For *CLASSes 1* to *3*, and *7* to *9*, there is the additional condition that the current channel interval is not more full than the adjacent channel interval (i.e., *f≤fa*). A wire segment which is in a full channel interval, or in a interval more full than the adjacent interval is put into *CLASS 10*.

CLASSes 1 to *3* contain switchable segments. *CLASS 4* contains essential cross-channel segments. *CLASS 5* contains non-essential cross-channel segments. *CLASS 6* contains top and bottom non-switchable same-row segments. *CLASSes 7* to *9* contain the normal non-switchable segments. Switchable segments are ranked higher than the other types of segments

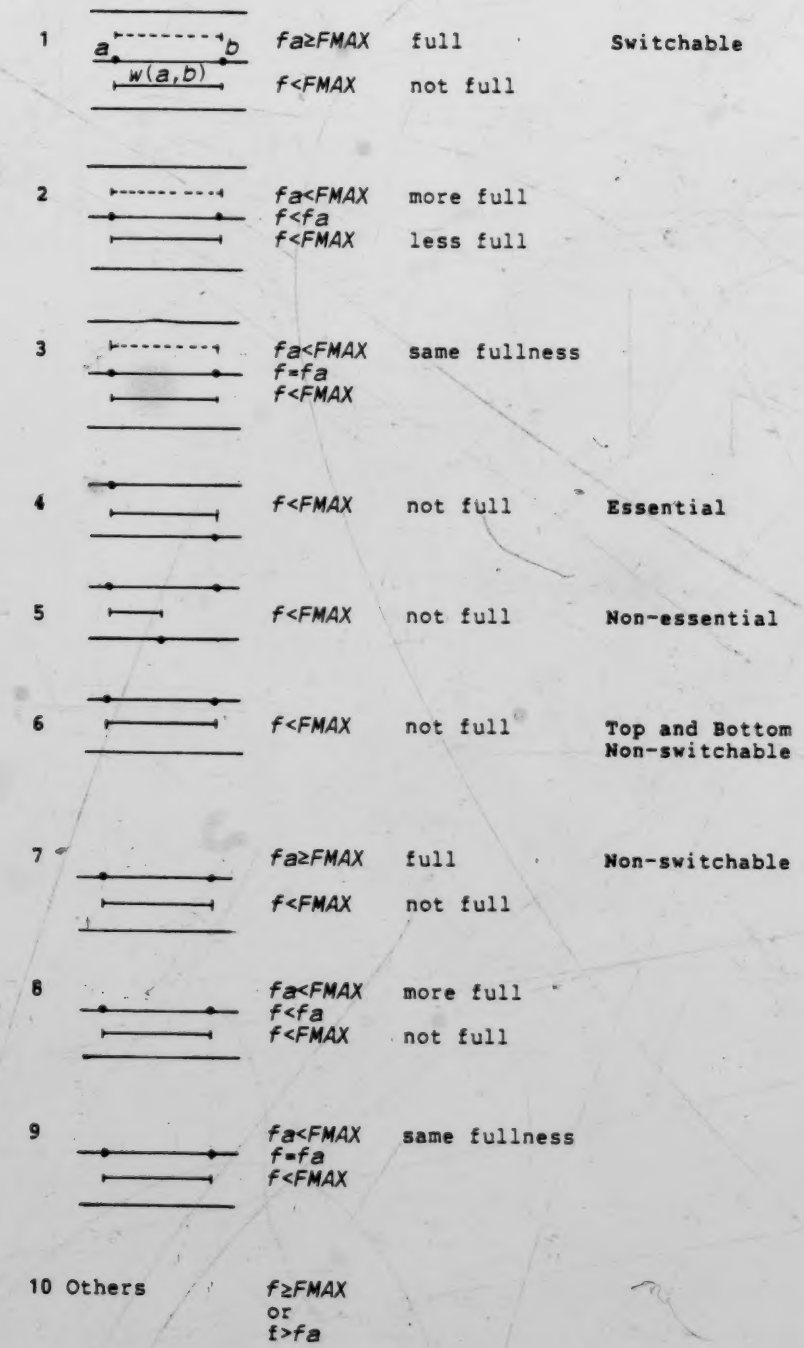


Fig. 4.10 Ranking of Segments in a Mutually Exclusive Set

because they allow us to manage the local densities. For example, in Fig. 4.11, terminals 1 and 2 can be connected by either the switchable segment $w_1(1,2)$ in C_1 or $w_2(1,2)$ in C_2. If the channel interval in C_1 is more full than C_2 (i.e., $f_1 > f_2$) then it is better to use $w_2(1,2)$.

Essential segments are said to be essential for the reason that they must be part of the tree which connects the net. Consider Fig. 4.12, terminals 1 and 2 can only be connected by segment $w(1,2)$ in channel C_2, whether there are other terminals of the same net above 1 or below 2. Since essential segments must be selected eventually, we rank them next to the switchable segments.

We rank the non-essential segments higher than the non-switchable segments based on the following observations. Consider Fig. 4.13 which shows a typical situation in which we have a choice between using a non-essential segment and a non-switchable segment. In Fig. 4.13a, terminals 1, 2 and 3

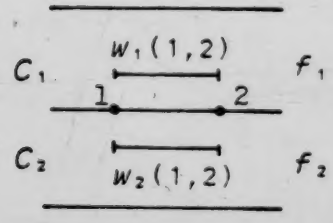


Fig. 4.11 Managing Local Densities Using Switchable Segments

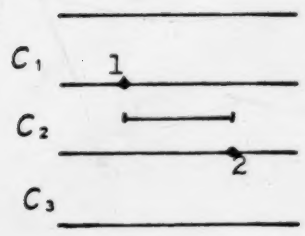


Fig. 4.12 Essential Segments Must Be Used

are connected by two non-essential segments. In Figs. 4.13b, and 4.13c, terminals 1, 2 and 3 are connected by one non-essential and one non-switchable segment. Fig. 4.13a is preferable because we only increase the local densities in C_1, and also use a shorter total wire length.

Among the non-switchable segments, the top and bottom non-switchable segments are ranked higher than other non-switchable segments. This is because terminals of a top/bottom non-switchable segment cannot be connected in the

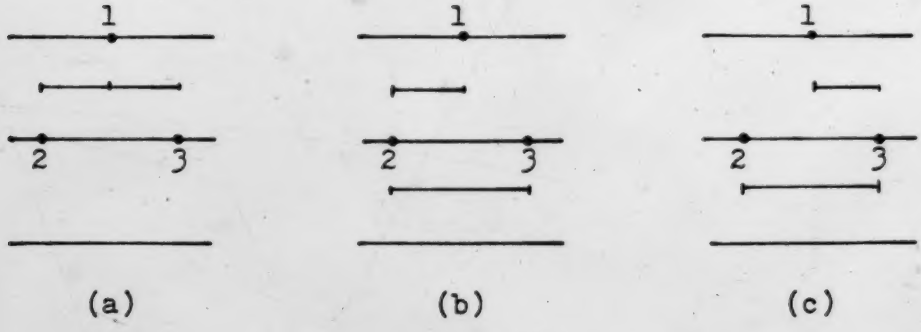


Fig. 4.13 Choosing Non-Essential Segment or Non-Switchable Segment

area above/below the top/bottom channel, whereas terminals of a normal non-switchable segment may be connected in the adjacent channel.

CLASS 1 is the highest ranked and *CLASS 10* is lowest. If the mutually exclusive set contains segments in *CLASSes 1* to *9*, then a segment is selected from the highest ranked class and added to S_2. If there are more than one segment in that class, then the following tie-breaking criteria are applied in order:

1. choose the segment with the minimum segment density
2. choose the segment with the maximum density in the adjacent channel interval
3. choose the shortest segment
4. choose the leftmost segment

We may use *FMAX* as a parameter to control the algorithm. If *FMAX*<1, then the algorithm will put wire segments only in less congested regions (with fullness<*FMAX*). These wire segments may help to make some unselected segments in the congested regions redundant. However, with *FMAX*<1, a state will be reached such that the segments left in S_1 are all in congested regions with $f \geq FMAX$. In that situation, all segments in the mutually exclusive set are in *CLASS 10*, and so no segments will be selected. Hence, our algorithm will increment *FMAX* by 0.1 after every cycle through the channels in which no segments can be selected. This will permit regions with gradually increasing fullness to be used for routing.

4.4.4 Global and Local Subnets

As tracks are being filled, we use the notion "subnets" to record what terminals of a net are connected. We define a *global subnet* n of net N to be a subset of N's terminals that are connected by selected segments in any channel of the layout.

Initially, we associate each terminal with an empty global subnet. Suppose a segment $w(a,b)$ is added to a track. Let $G(a)$ denote the global subnet which contains terminal a, and let $G(b)$ denote the global subnet which contains terminal b. If $G(a)=G(b)=\phi$, then we create a subnet containing a and b, i.e., $G(a)\leftarrow\{a,b\}$ and $G(b)\leftarrow\{a,b\}$. If $G(a)\neq\phi$ and $G(b)=\phi$, then we add b to $G(a)$, i.e., $G(a)\leftarrow G(a)\cup\{b\}$ and $G(b)\leftarrow G(a)$. If $G(a)\neq\phi$, $G(b)\neq\phi$, and $G(a)\neq G(b)$, then we merge the subnets by adding the terminals of the smaller subnet to the larger subnet. That is, if $|G(a)|>|G(b)|$, then $G(a)\leftarrow G(a)\cup G(b)$, and $G(b)\leftarrow G(a)$. A segment $w(a,b)$ is redundant if a and b belong to the same global subnet. After global routing, all terminals of net N must belong to the same global subnet G.

It is insufficient to consider just global subnets because we have to generate individual net-lists for each channel so that the channels can be routed independent of each other by the detailed router. For example, in Fig. 4.14, terminals 1 and 2 are connected by $w_1(1,2)$ in channel C_1 and hence they belong to the same global subnet. However, 1 and 2 should not be connected by wire segment

$w_2(1,2)$ in channel C_2. Hence we have to introduce the idea of *local subnets* to indicate whether two terminals in a channel are connected by a segment in that channel. We define a *local subnet* l of net N in channel C to be a subset of N's terminals that are in C and are connected by selected wire segments in S_2 of C.

Whenever a segment $w(a,b)$ is selected in a channel C, besides updating the global subnets as described above, we also update the local subnets in C. The updating of local subnets is similar to the updating of global subnets except that we only consider the terminals and local subnets in C. For example, in channel C_1 of Fig. 4.15, terminals 1 and 3 belong local subnet l_1, terminals 2 and 4 to local subnet l_2, $l_1 \neq l_2$. If segment $w(3,4)$ is selected, then l_1 and l_2 are merged so that terminals 1, 2, 3, and 4 belong to the same local subnet l in channel C_1. On the other hand, terminals 3 and 4 still belong to different local subnets in channel C_2. After global routing, the local subnets in a channel C

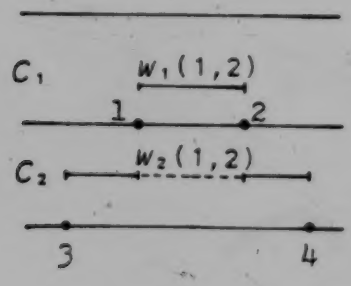


Fig. 4.14 Need of Local Subnets

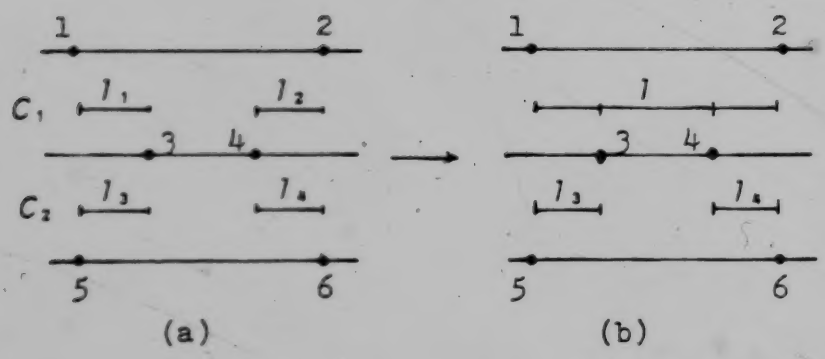


Fig. 4.15 Local Subnet

specify the net-list of C so that C can be routed by the channel router independent of other channels.

4.5 Improvement of Initial Solution by Segment Swapping

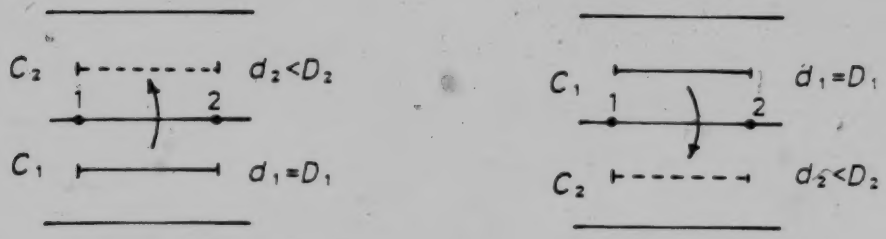
The initial solution constructed by the track-filling algorithm can be improved to reduce the total channel densities. We use a 4-pass improvement step featuring *segment swapping*.

Segment swapping is the process of swapping a switchable same-row segment from one channel to the next channel in an attempt to reduce the density of the channel to which the segment was originally assigned. We illustrate the idea of segment swapping with Fig. 4.16a. Suppose channel C_1 has density D_1, channel C_2 has density D_2. The switchable same-row segment $w(1,2)$ which connects terminals 1 and 2 has segment density d_1 in channel C_1. The maximum local density between terminals 1 and 2 in channel C_2 is d_2. If $d_1=D_1$ and $d_2<D_2$, then we may swap the segment $w(1,2)$ from

C_1 to C_2 without increasing the density of C_2, while the local densities of the same interval in C_1 decrease by 1, and so the density of C_1 may decrease by 1. The case in Fig. 4.16a in which a segment is swapped from a channel below the terminals to a channel above the terminals is called an *upward swap*. Similarly, a *downward swap* is one in which a segment is swapped from the channel above to the channel below the terminals as shown in Fig. 4.16b.

To attempt to reduce the density of a channel, we process the channel in the following way. We examine all the selected switchable same-row segments in the channel from left to right. If a segment satisfies the above swapping condition, then we swap it to the other channel, update the densities of both channels, and proceed to the next segment.

The improvement step we use consists of four passes through the chip. In each pass, we process all the channels in sequence either from top to bottom or from bottom to top,



(a) Upward Swap (b) Downward Swap

Fig. 4.16 Segment Swapping

and perform only upward or downward swaps (Fig. 4.17). The four passes are:

1. Pass 1 - process channels from top to bottom, perform downward swaps
2. Pass 2 - process channels from bottom to top, perform upward swaps
3. Pass 3 - process channels from top to bottom, perform upward swaps
4. Pass 4 - process channels from bottom to top, perform downward swaps

We perform swaps in only one direction during each pass to avoid segments being swapped back and forth in the same pass. For example in Fig. 4.18, if we allow swaps in both directions during pass 1, then the segment may be swapped from C_1 to C_2 when C_1 is being processed, and swapped back to C_1 when C_2 is being processed. By processing the channels in order and swapping wires in only one direction during

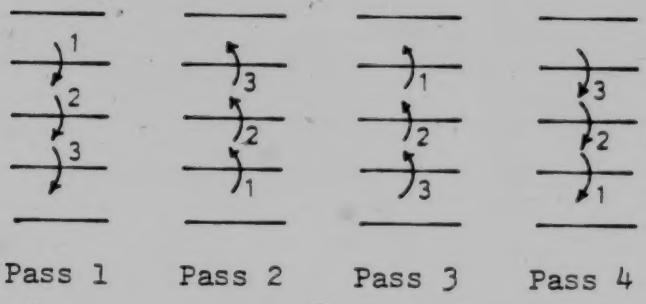


Fig. 4.17 Four Pass Segment Swapping

each pass, we systematically distribute the densities. Pass 1 shifts wires from top to bottom, pass 2 shifts wires from bottom to top.

Consider Fig. 4.19a. We cannot swap segment $w(3,4)$ from C_1 to C_2 because $d_2=D_2$ in C_2. However, in channel C_2 there may be segments, such as $w(1,2)$, which can be swapped to C_3, thus creating room for $w(3,4)$ to be swapped to C_2. Hence if we process C_2 before C_1 and swap $w(1,2)$ to C_3 first, then we would be able to swap $w(3,4)$ to C_2. Pass 3 is intended to handle this type of situation. Similarly, Pass 4 is intended to handle the analogous situation shown in Fig. 4.19b.

It should be pointed out that this 4-pass segment swapping improvement step is applicable to initial global routing solutions produced by any other global routing algorithms; it is not intended only for our track-filling algorithm.

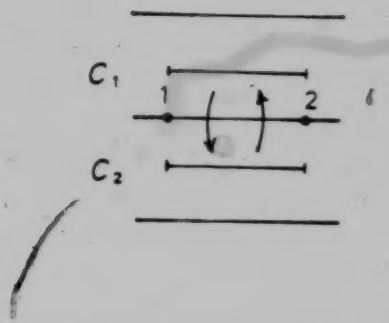


Fig. 4.18 Segment Swapped Back and Forth

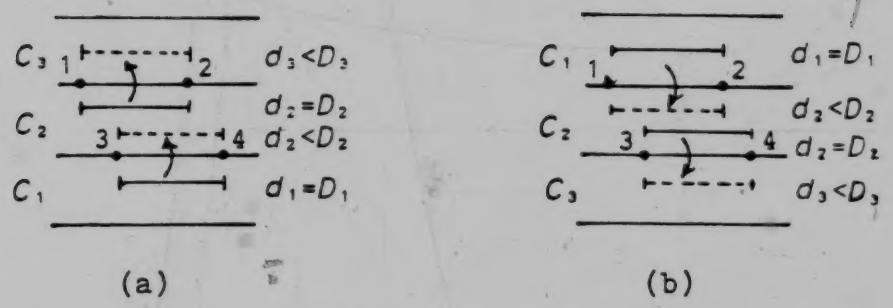


Fig. 4.19 Purpose of Pass 3 and Pass 4

Chapter 5

Experiments and Evaluation

In this chapter, we present the results and observations obtained from the experiments that were performed on the track-filling global algorithm. The main objectives of our experimentation are: (1) to investigate the parameters of our algorithm, (2) to evaluate the performance of the algorithm, and (3) to observe the behaviour of the algorithm.

5.1 The Routing Programs and Test Circuits

The track-filling global router and the gridless channel router described in the previous two chapters have been implemented in the C-language on a VAX 11/780 under the Berkeley Unix 4.2 operating system. The input to the global router program is a file produced by the placement program describing the relative positions of the cells in the rows. The global router accesses the circuit database to get the net-list of the circuit, the cell dimensions and the terminal positions. The output after global and detailed routing is a checkplot of the routed circuit as shown in Fig. 5.1. The global router was tested using 15 circuits of size ranging from about 100 cells to 1800 cells. Table 5.1 gives the statistics of the circuits. Since the objective of the track-filling algorithm was to minimize the total

channel density, ΣDf, we will use ΣDf to evaluate the quality of the results.

5.2 Investigation of the Parameters

Our objective here is to determine if ΣDf depends on the parameters. If so, we want to determine how these parameters should be used to yield the best algorithm. The parameters are: (1) density estimate methods, (2) *FMAX*, and (3) channel ordering. The three density estimates described in Chapter 4 were tested. To investigate how *FMAX* influences ΣDf, ten values of *FMAX*, 0.1, 0.2, ..., 0.9 and 1.0 were used. For the ordering of the channels, we tested both ascending and descending orders of the initial channel density estimates. The track-filling global router was used to route the 15 circuits using all combinations of the above parameter variations. The experimental results are summarized and presented in Table 5.2. In the table, each entry represents the sum of the ΣDfs of the 15 circuits after global routing. The averages over all values of *FMAX* were also computed.

In Chapter 4, we described three density estimate methods that may be used to estimate fullness during global routing. Recall that d_1 and D_1 are computed by assuming that the unselected segments in S_1 and the selected segments in S_2 will be routed in the channel, d_2 and D_2 are computed by assuming that only the selected segments in S_2 will be routed in the channel, and $d_3=(d_1+d_2)/2$, $D_3=(D_1+D_2)/2$. In

method 1, we use d_1 and D_1 to compute fullness f_1, i.e., $f_1=d_1/D_1$. Similarly, in method 3, $f_3=d_3/D_3$. Since D_2 is zero initially, we use D_1 instead of D_2 as the denominator in method 2, i.e., $f_2=d_2/D_1$. Our experimental results as shown in Table 5.2 indicate that estimate method 1 yields slightly smaller average ΣDf than does method 3, although the difference is very small. Method 2 yields higher ΣDfs than methods 1 and 3.

Our experimental results do not show any signs that ΣDf depends on *FMAX*. As shown in Table 5.2, there are some small variations in the final total densities for different *FMAX*s. However, the variations are not significant and they do not follow any pattern. Hence, we conclude that for the test circuits we used, changing *FMAX* has no effect on the quality of the routing.

In Chapter 4, we conjectured that the order in which the channels are cycled would not have a large effect on the results. To test this, we tested both ordering the channels by ascending and descending order of their initial estimated densities. The experimental results show that for density methods 1 and 3, there is no significant difference between the two ordering methods. This is due to the parallel nature of our algorithm in that it puts one track through every channel before any channel receives a second track. For density method 2, cycling the channels in descending order seems to be better.

5.3 Evaluation of the Algorithm

Our second major objective of experimentation is to evaluate how well our algorithm performs compared against existing algorithms. Since the literature does not contain enough data for comparison purposes, we had to implement a sequential tree-type global router which finds a minimum spanning tree for each net in sequence. The algorithm we implemented has similar characteristics with the global routers in [Sec84], [Aos83], and [Wie84]. Details of the implemented algorithm are given in Appendix II. We used this tree-type global router to route the 15 circuits, and the results are presented in Table 5.3. In this table, the data reported for the track-filling algorithm were obtained using the density estimate method 1, $FMAX=1$, and ordering the channels by decreasing channel densities.

The experimental results shown in Table 5.3 indicate that our algorithm performs slightly better in terms of the final total channel densities. For most of the circuits, our algorithm results in smaller ΣDf than the tree-type router does. The difference becomes more significant as the circuit size increases (especially for circuits with more than 400 cells).

The times reported in Table 5.3 are VAX 11/780 CPU seconds. By plotting log_{10}(number of terminals) against log_{10}(time), and calculating the slope of the fitted straight line (Fig. 5.2), we empirically determined that the time complexity of the track-filling algorithm is about

$O(T^{1.3})$, where T=number of terminals. Similarly, the time complexity is $O(E^{1.2})$, where E=total number of canonical wire segments (Fig. 5.3). We expect the time complexity of the tree-type global router to be about $O(E\log_2 e)$, where e=average number of canonical wire segments per net (see Appendix II). We plotted $\log_{10}(E\log_2 e)$ against $\log_{10}$(time) for the tree-type algorithm and determined the slope to be 1.1 (Fig. 5.4). We also plotted $\log_{10}(E\log_2 e)$ against $\log_{10}$(time) for our algorithm and discovered that the slope of the straight line is also 1.1 (Fig. 5.5). This shows that the time complexity of the track-filling algorithm is about the same as the tree-type algorithm for circuits of reasonable sizes which we have tested.

In another experiment, the circuits were routed without global routing. We used the detailed router to route the channels one at a time from top to bottom. As shown in Table 5.3, the total channel densities are about 10% higher than if global routing is used. Hence, the importance of using a global router is demonstrated. In [Sec84], it was also reported that with the TimberWolf global router, the number of wiring tracks required is about 10 to 15% less than if no global routing is used. However, the TimberWolf global router requires much more computer time than our track-filling algorithm.

We tested the effectiveness of our track-filling heuristics in another experiment. In this experiment, we used a method similar to the "left-edge" algorithm [Hash71]

to select the wire segment from the mutually exclusive set. In the left-edge method, we always select the leftmost wire segment in the set, without applying the wire selection heuristics. As shown in Table 5.3, this left-edge method produced better results than with no global routing at all, but they are still about 5% worse than if we use the wire selection heuristics. This demonstrates the effectiveness of our heuristics.

We also tested the effectiveness of the 4-pass improvement step, and the results are shown in Table 5.4. The results show that the improvement step reduces the ΣDfs by about 2% on average, while using about 50% more time. However, by applying this 4-pass improvement step to the initial solution produced by the tree-type router, we discovered that the ΣDf is reduced to below that of the track-filling algorithm in many circuits.

5.4 Some Observations

Besides performing the experiments described above, we also made several observations on the results of the experiments. In the following paragraphs, we describe these observations.

For every circuit, we computed $D'*E'/E$, where D'=the total initial channel densities before global routing assuming all wire segments are routed in each channel, E'=the total number of wire segments selected in global routing, and E=the total number of canonical wire segments.

As shown in Table 5.5, for all circuits but circuit 2, $D'*E'/E$ is consistently lower than ΣDf. Hence, we suggest that $D'*E'/E$ may be used as a rough estimate of the number of wiring tracks required. This estimate may be used by a placement algorithm to evaluate how good a placement is without actually performing routing. We feel that $D'*E'/E$ is a reasonable estimate of ΣDf because of the following assumptions. Assume that before global routing, all the E segments are evenly distributed in every channel and the total channel density is D'. Suppose that after global routing, the E' selected wire segments are also evenly distributed. By proportion, we estimate the final total channel density to be $D'*E'/E$. However, in practice there are more segments in the center of the chip, and hence the actual ΣDf obtained is higher than $D'*E'/E$.

We also calculated the fraction of the chip area that are used for wiring in the circuits. As shown in Table 5.5, we observe that as the chip size increases, the fraction of the chip area used for wiring increases. For our circuits, the percentage of wiring area increased from about 60% to over 80%. This demonstrates the need for good global and detailed routers to reduce the routing area.

We also observe how the three density estimates D_1, D_2, and D_3 vary as the algorithm proceeds. Fig. 5.6 shows a typical plot of ΣD_1, ΣD_2, ΣD_3 against the number of cycles through the channels. As predicted in Chapter 4, ΣD_1 decreases and ΣD_2 increases monotonically towards ΣDf, and

ΣD_3 is closer to ΣDf.

To observe how the tracks are being filled, the segments added to each track were plotted as the program proceeded. Fig. 5.7 shows the tracks of circuit 4 obtained with *FMAX*=1, Fig. 5.8 shows the tracks obtained with *FMAX*=0.7, and Fig. 5.9 shows the tracks obtained using the left-edge method. We observe that in Fig. 5.7 and Fig. 5.8, there are "holes" in the tracks. These holes correspond to the situation when no segment is selected from the mutually exclusive set either because the channel interval is full or more full than the adjacent channel interval. As the algorithm proceeds, some unselected segments in these intervals become redundant, and so these intervals are no longer full, or become less full than the adjacent channel. The occurrence of these holes are more prominent in Fig. 5.8. This is because initially intervals with fullness≥0.7 are considered full, and hence no segments are selected. As *FMAX* is incremented, the algorithm begins to select wire segments in these intervals, and so there are less holes in the later tracks. Note that in Fig. 5.9, the tracks seem to be quite densely packed because a segment closest to the previous segment in the track is always chosen. However, the wire segments selected by our track-filling algorithm can be packed into less tracks than those selected by the left-edge method.

A plot of the density profiles before and after global routing is shown in Fig. 5.10. It can be seen that the final

density profile is more flat than the initial profile, especially in the center of the chip where congestions are likely to occur before global routing. This indicates that the wires are distributed fairly evenly such that congested spots are avoided. Although the general contour of the final density profile follows the initial profile, congested spots in the original profile were avoided. Note also that the final densities in the top and bottom channels are closer to the initial densities. This is because terminals on the top and bottom rows have no chance of being connected in an alternate channel, and hence fewer segments in the top and bottom channel will become redundant. The checkplot for circuit 4 which we have considered in Figs. 5.6 to 5.10 is given in Fig. 5.11.

Table 5.1 Statisties of the Test Circuits

Circuit	Cells	Rows	*T*	*N*	*T/N*	*E*	*E/N*
1	86	6	282	76	3.7	245	3.2
2	88	5	246	93	2.6	224	2.4
3	116	6	375	119	3.2	349	2.9
4	162	7	507	156	3.3	542	3.5
5	218	6	763	217	3.5	786	3.6
6	258	6	762	272	2.8	685	2.5
7	306	7	1144	306	3.7	1140	3.7
8	392	8	1400	408	3.4	1340	3.3
9	400	8	1512	442	3.4	1559	3.5
10	446	8	1674	523	3.2	1922	3.7
11	578	10	2275	612	3.7	2483	4.1
12	631	11	2376	570	4.2	2718	4.8
13	802	12	3003	766	3.9	3327	4.3
14	1352	15	4298	1266	3.4	5030	4.0
15	1798	15	5556	1728	3.2	6613	3.8

T = number of terminals
N = number of nets
T/N = average number of terminals per net
E = number of canonical wire segments
E/N = average number of canonical wire segments per net

Table 5.2 Experimental Results Obtained with Variations of Parameters

	Estimate 1		Estimate 2		Estimate 3	
FMAX	D	A	D	A	D	A
.1	2603	2600	2739	2746	2607	2607
.2	2604	2607	2739	2747	2611	2611
.3	2594	2599	2739	2747	2607	2607
.4	2598	2593	2739	2747	2609	2600
.5	2601	2596	2739	2747	2612	2619
.6	2593	2598	2739	2747	2613	2609
.7	2590	2591	2739	2747	2607	2607
.8	2593	2601	2739	2747	2613	2613
.9	2592	2598	2739	2747	2609	2615
1.0	2594	2607	2739	2747	2612	2614
Ave.	2596	2598	2739	2747	2610	2610

D = descending order
A = ascending order

Table 5.3 Experimental Results Obtained With Track-Filling Algorithm, Tree-Type Algorithm, No Global Routing, and Left-Edge Method

Circuit	Track-Fill ΣDf	Time (sec)	Min-span-tree ΣDf	Time (sec)	No global ΣDf	Left-edge ΣDf
1	48	3.3	48	3.1	50	51
2	30	2.7	30	2.3	31	30
3	57	4.6	55	4.3	64	58
4	59	6.7	60	7.5	66	61
5	93	12.3	95	12.4	99	96
6	93	10.8	92	10.0	102	97
7	120	18.6	120	20.2	136	128
8	168	26.1	172	23.2	172	169
9	161	29.4	164	31.1	170	166
10	180	36.7	184	39.1	200	188
11	182	52.7	186	56.8	209	194
12	242	51.3	240	58.6	256	250
13	298	71.4	299	81.5	324	320
14	370	109.4	381	101.2	413	392
15	493	172.4	508	149.7	542	509
Total	2594		2634		2834	2709

Table 5.4 Experimental Results Obtained With 4-Pass Improvement Step

Circuit	Track-Fill ΣDf		Time (sec)		Min-Span-Tree ΣDf	
	NI	I	NI	I	NI	I
1	48	48	3.3	4.3	48	48
2	30	28	2.7	3.7	30	29
3	57	56	4.6	6.2	55	55
4	59	58	6.7	8.9	60	58
5	93	91	12.3	16.8	95	92
6	93	93	10.8	15	92	90
7	120	119	18.6	27	120	116
8	168	167	26.1	36.9	172	167
9	161	160	29.4	42.2	164	158
10	180	173	36.7	52.7	184	175
11	182	181	52.7	74.6	186	179
12	242	238	51.3	75.2	240	233
13	298	291	71.4	100.9	299	290
14	370	362	109.4	152.8	381	366
15	493	479	172.4	244	508	483
Total	2594	2544			2634	2540

NI = initial solution without improvement
I = with improvement

Table 5.5 Correlation of ΣDf with $D'*E'/E$, and Increase of Wiring Area

Circuit	Cells	Rows	E	E'	D'	ΣDf	$D'*E'/E$	W/A
1	86	6	245	206	55	48	46	63
2	88	5	224	154	43	30	30	58
3	116	6	349	256	72	57	53	67
4	162	7	542	363	85	59	57	65
5	218	6	786	554	116	93	81	76
6	258	6	685	503	121	93	88	77
7	306	7	1140	840	158	120	117	78
8	392	8	1340	993	186	168	138	81
9	400	8	1559	1134	190	161	139	81
10	446	8	1922	1258	250	180	163	82
11	578	10	2483	1732	239	182	167	81
12	631	11	2718	1830	304	242	204	78
13	802	12	3327	2273	378	298	257	84
14	1352	15	5030	3038	518	370	311	84
15	1798	15	6613	3839	727	493	422	87

E = number of canonical wire segments
E' = number of wire segments used to connect all nets
D' = total channel density if all canonical wire segments are used
ΣDf = total channel density after global routing
W/A = percentage of chip area used for wiring

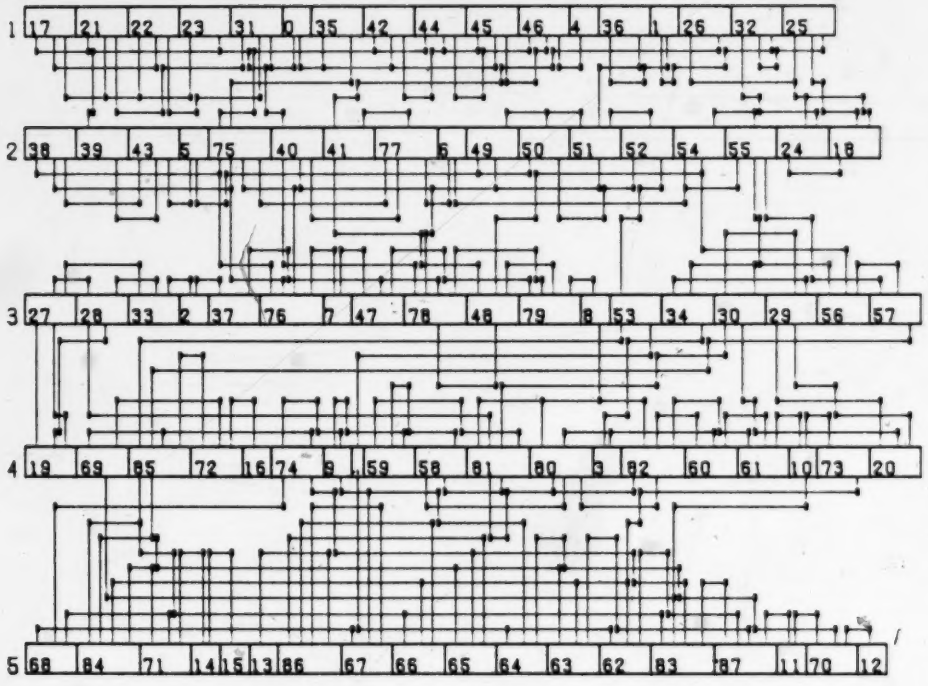


Fig. 5.1 Checkplot of a Circuit After Routing

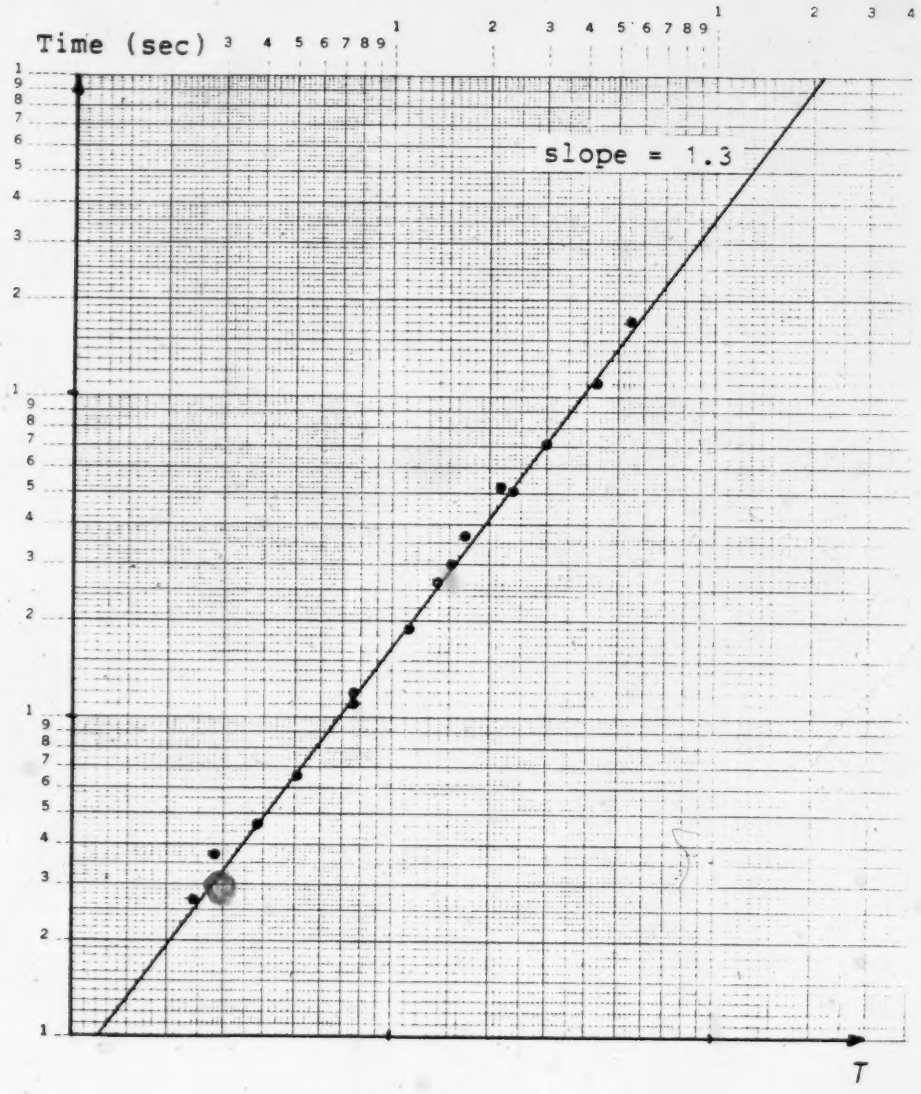


Fig. 5.2 log_{10}(number of terminals) vs log_{10}(time) for Track-Filling Algorithm

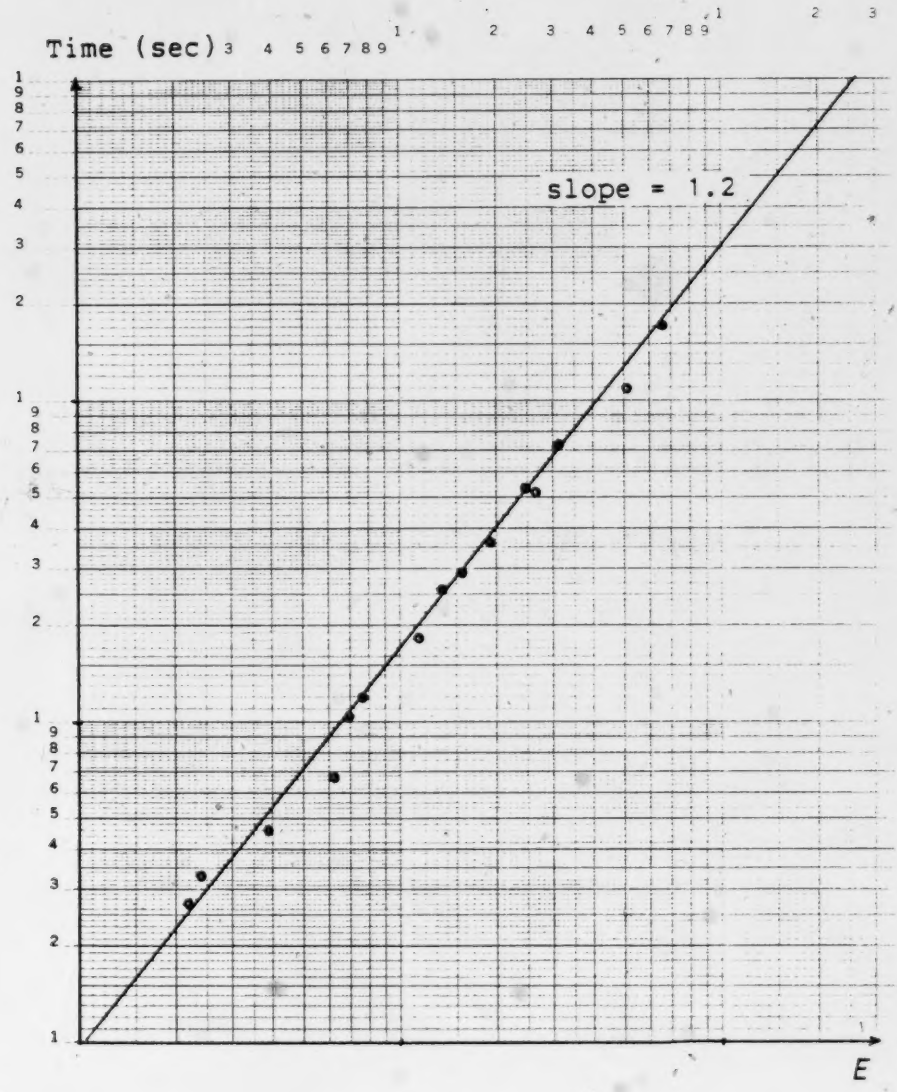


Fig. 5.3 log_{10}(number of canonical wire segments) vs log_{10}(time) for Track-Filling Algorithm

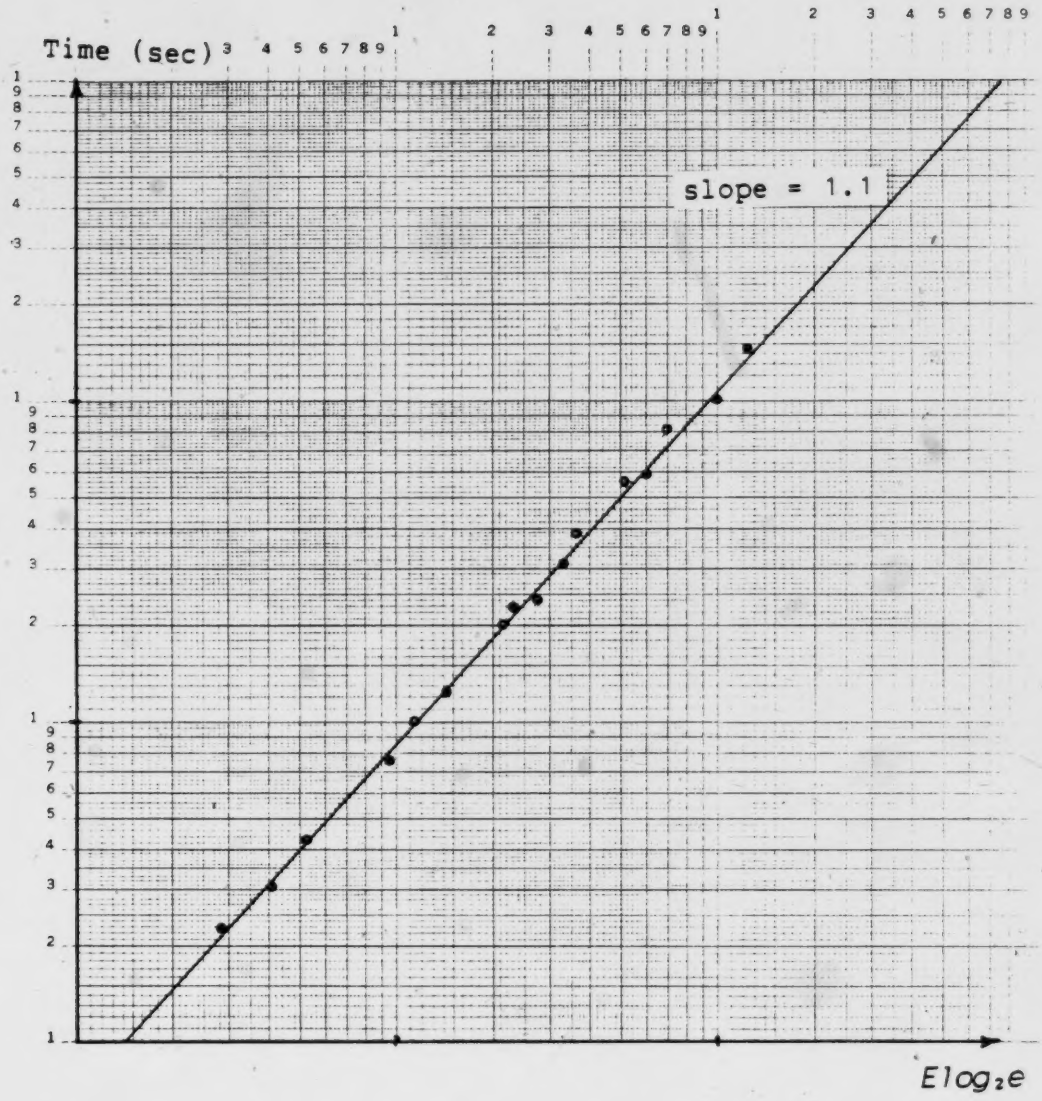


Fig. 5.4 $log_{10}(Elog_2e)$ vs log_{10}(time) for Tree-Type Algorithm

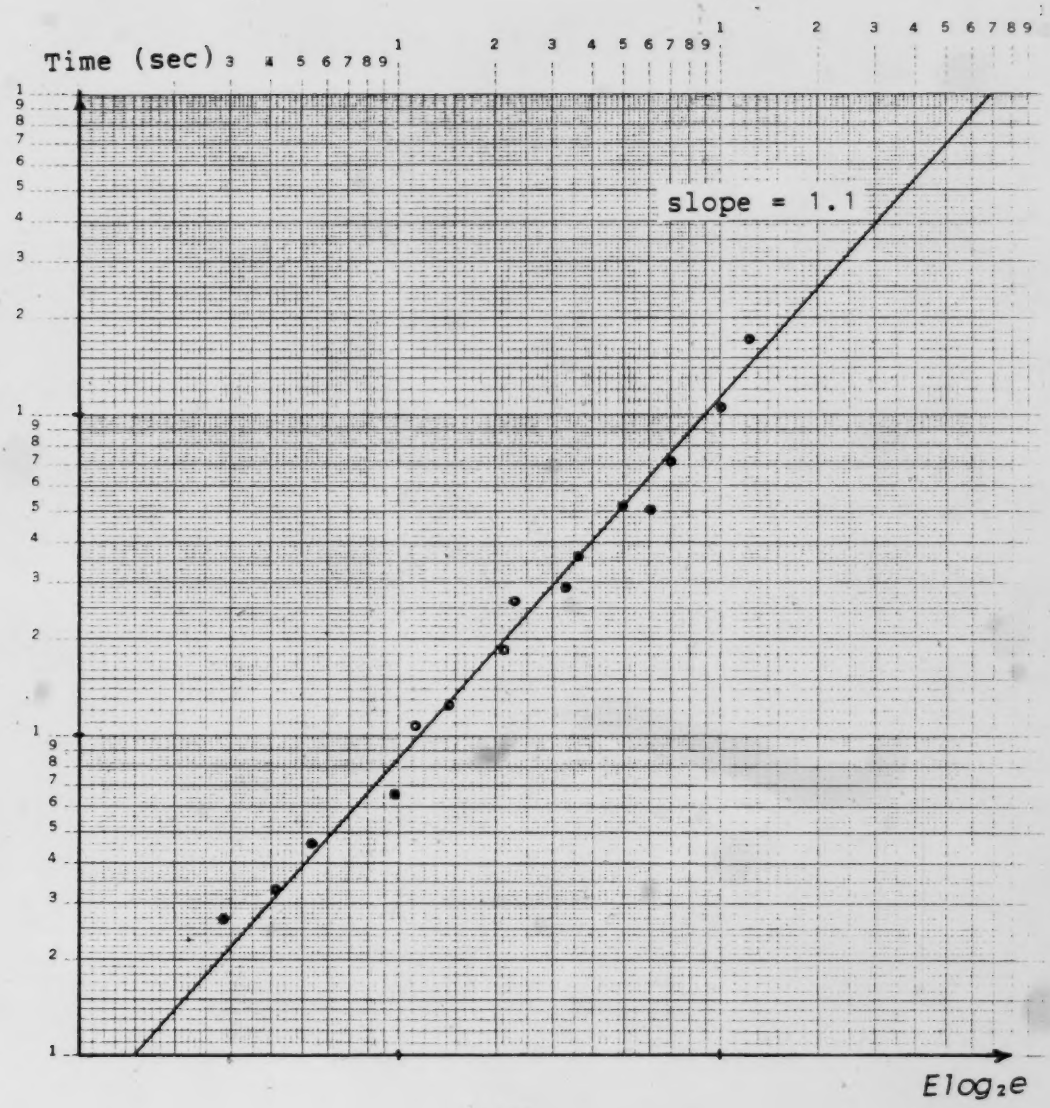


Fig. 5.5 $\log_{10}(E\log_2 e)$ vs $\log_{10}$(time) for Track-Filling Algorithm

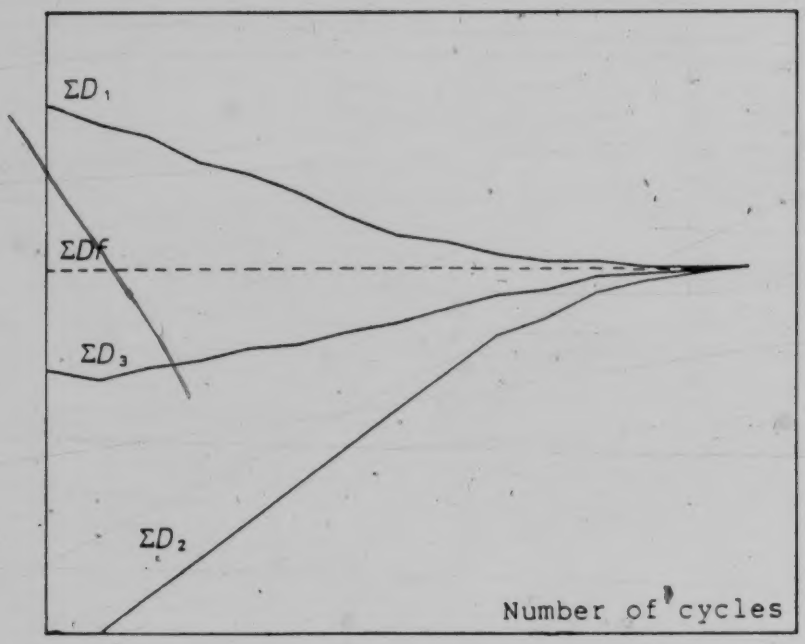


Fig. 5.6 Plot of the Three Density Estimates During Global Routing

Fig. 5.7 Tracks Obtained With *FMAX*=1.0

Fig. 5.8 Tracks Obtained With *FMAX*=0.7

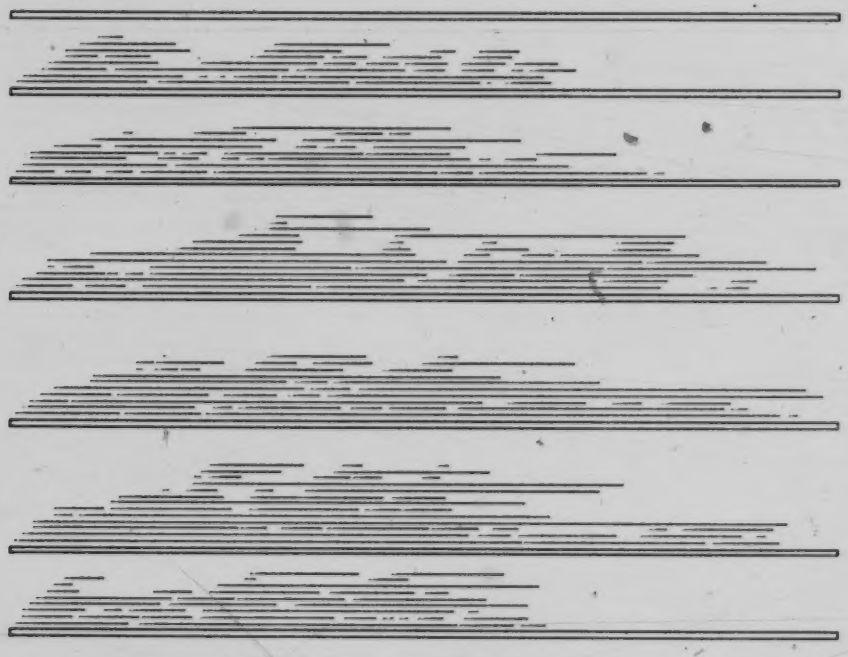


Fig. 5.9 Tracks Obtained Using Left-Edge Method

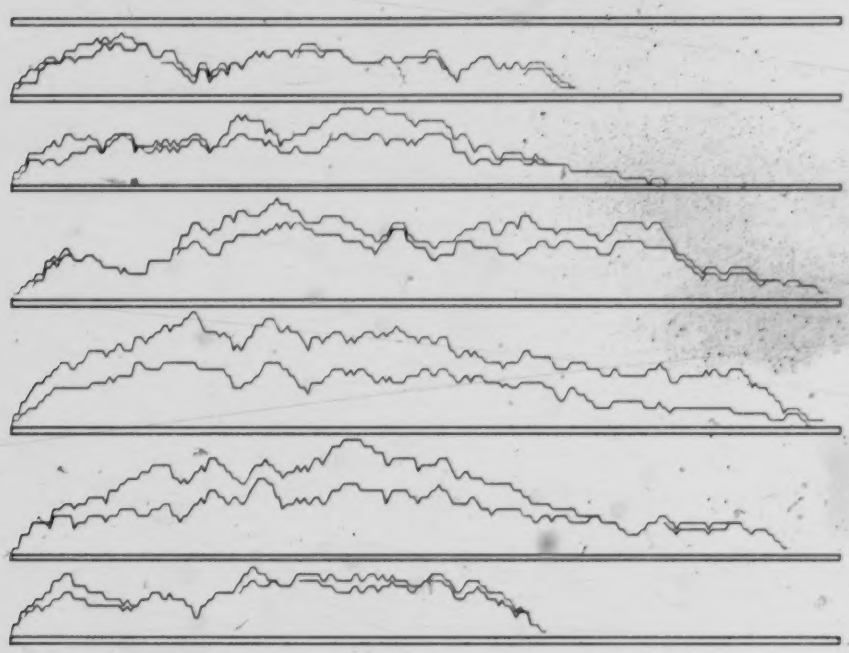


Fig. 5.10 Initial and Final Density Profiles

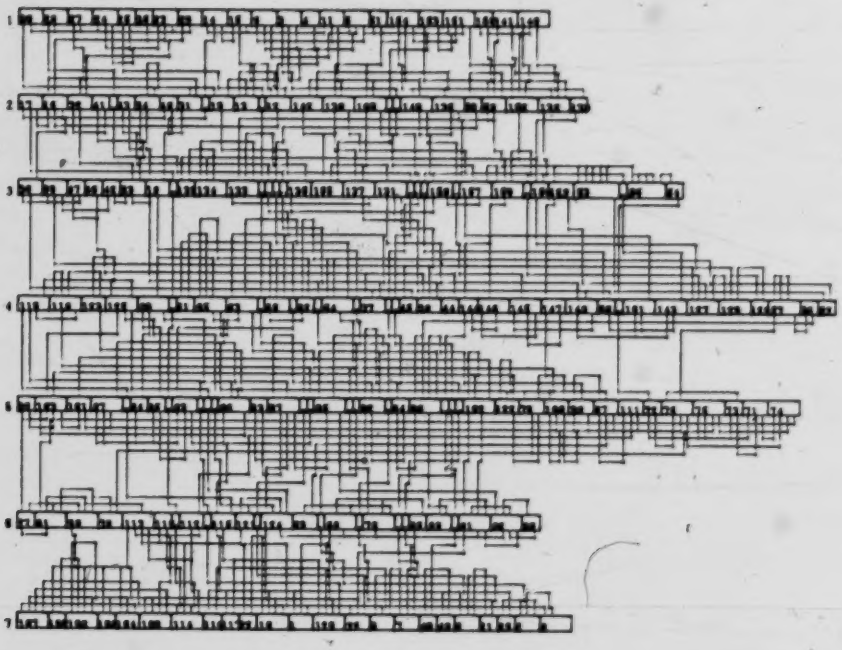



Fig. 5.11 Checkplot of Circuit 4 After Routing

Chapter 6

Conclusions and Future Research

In this thesis, we have looked at the global routing and detailed routing problems of standard-cell ICs. The main contribution of this work is the development of a new parallel standard-cell global routing algorithm which uses a track-filling approach. The basic principle of the track-filling approach is to build a track of wire segments in each channel. The wire segments are selected from the canonical set of wire segments of the channel and added to the track from left to right. The notions of fullness and segment types are used in the wire segment selection process. Fullness is used to estimate the relative density of the final routing in a channel region. The wire selection process categorizes the canonical wire segments into ranked classes according to fullness and segment types. The highest ranked segment is selected and added to the track. This track-filling approach is different from the conventional sequential tree-type approach because it does not process the nets one at a time, hence making the algorithm independent of the order in which the nets are processed.

The track-filling algorithm has been implemented and was tested with 15 realistic circuits of reasonable sizes. Experimental results show that our track-filling global routing algorithm performs slightly better on these 15

circuits than a tree-type global router which we implemented for comparison purposes. The time required by our algorithm is empirically measured to be about the same as that of the tree-type router. The tree-type approach has been in use for a long time, and there have not been any recent innovative improvements to this type of algorithm. On the other hand, this new track-filling approach we propose is applied to perform global routing for the first time. We feel that our track-filling algorithm has potential to be improved by future research.

Numerous possible improvements and extensions can be made to the track-filling algorithm in the future. Since the track-filling idea is similar to that used by many channel routing algorithms, it seems likely that the track-filling algorithm can be modified so that it can perform global and detailed routing simultaneously. This implies that the vertical positions of the wire segments in the tracks will have to be considered too. Also, the wire selection process would have to consider vertical constraints in the channels and try to avoid formation of cyclic constraints. Another improvement which could be made would be to introduce the notion of critical nets into the algorithm. Critical nets are nets for which signal propagation delay is important to the performance of the circuit, e.g, clock signal. It is important to connect critical nets using the shortest wires. The track-filling algorithm may be modified to route the critical nets first using the shortest wires before applying

the track-filling process to the other nets. The track-filling algorithm can also be adapted to global route row-cell style gate-array ICs. However, it is not apparent how this algorithm can be extended to handle global routing of general-cell ICs.

In this thesis we have also looked at the gridless channel routing problem that arises in our standard-cell system. A minor contribution of this research is that we have modified an existing gridded channel router so that it can route channels in which the terminals do not necessarily lie on vertical grid lines. This allows the cells to be designed in a more area-efficient manner because their widths and terminal spacings are not restricted to be multiples of the minimum inter-wire spacing. In the future, this gridless router can be further modified to become truly gridless so that the horizontal wires are not restricted to line on horizontal grid lines. Another possible improvement is that a post-processing step may be used to maximize the amount of metal wires by converting polysilicon wires, if possible, into metal wires and reducing the number of contacts (Fig. 6.1). It is desirable to use as much metal for wiring as possible because metal is a better conductor than polysilicon.

Besides the improvements to the global and detailed router suggested above, our system needs further work before it is completed. For example, more cells need to be designed for our cell library, and the routing of signals to I/O pads

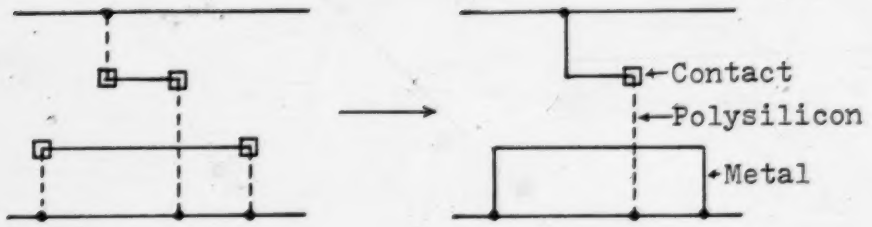


Fig. 6.1 Maximize the Amount of Metal Used in Routing

needs to be considered. Also a piece of software is required to generate the geometric patterns after the IC has been placed and routed.

We have demonstrated that as the number of cells in a standard-cell IC increases, the fraction of the chip area used for wiring increases dramatically. Therefore it is important to have a good global router and a good detailed router that can route the chip in minimum area. With our global routing algorithm, we can route a chip in about 10% less tracks than without global routing. We feel that the work reported in thesis has made some progress in minimizing routing area for standard-cell ICs.

Bibliography

[Aos83] K. Aoshima, E. S. Kuh, "Multi-Channel Optimization in Gate-Array LSI Layout", *Proceedings of the 1983 IEEE International Symposium on Circuits and Systems*, 1983, pp. 1005-1008.

[Bur83] M. Burstein, R. Pelavin, "Hierarchical Channel Router", *Proceedings of the 20th Design Automation Conference*, 1983, pp. 591-597.

[Bur85] M. Burstein, "Channel Routing", *IBM Research Report RC 10973*, 1985.

[Che77] K. A. Chen, M. Feuer, K. H. Khokani, N. Nan, S. Schmidt, "The Chip Layout Problem: An Automatic Wiring Procedure", *Proceedings of the 14th Design Automation Conference*, 1977, pp. 298-302.

[Deu76] D. Deutsch, "A 'Dogleg' Channel Router", *Proceedings of the 13th Design Automation Conference*, 1976, pp. 425-433.

[Gar79] M. R. Garey, D. S. Johnson, *Computers and Intractability*, San Franscio, W. H. Freeman, 1979.

[Ham84] G. T. Hamachi, J. K. Ousterhout, "A Switchbox Router With Obstacle Avoidance", *Proceedings of the 21th Design Automation Conference*, 1984, pp. 173-179.

[Hash71] A. Hashimoto, J. Stevens, "Wire Routing by Optimal Channel Assignment Within Large Apertures", *Proceedings of the 8th Design Automation Workshop*, 1971, pp. 155-169.

[Hass82] J. E. Hassett, "Automatic Layout in ASHLAR: An Approach to the Problems of General Cell Layout For VLSI", *Proceedings of the 19th Design Automation Conference*, 1982, pp. 777-784.

[Hig69] D. W. Hightower, "A Solution to the Line-Routing Problem on the Continuous Plane", *Proceedings of the 6th Design Automation Workshop*, 1969, pp. 1-24.

[Jen84] P. I. Jennings, S. L. Hurst, A. McDonald, "A Highly Routable ULM Gate-Array and its Automated Customization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-3, No. 1, Jan. 84, pp. 27-40.

[Ker73] B. W. Kernighan, D. G. Schweikert, G. Persky, "An Optimum Channel-Routing Algorithm for Polycell Layouts of Integrated Circuits", *Proceedings of the 10th Design Automation Workshop*, 1973, pp. 50-59.

[Kim83] S. Kimura, N. Kubo, T. Chiba, I. Nishioka, An Automatic Routing Scheme For General Cell LSI", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No. 4, Oct. 1983, pp. 285-292.

[Kir82] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, May 1983, pp. 671-680.

[Kol77] K. W. Koler, U. Lauther, "The Siemens-AVESTA-System for Computer-Aided-Deisgn of MOS-Standard Cell Circuits", *Proceedings of the 14th Design Automation Conference*, 1977, pp. 153-157.

[Kru56] J. B. Kruskal, Jr, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Porblem", *Proc. American Mathematical Society*, Vol. 7, No. 1, 1956, pp. 48-50.

[LaP80] A. S. LaPaugh, "Algorithms for Integrated Circuit Layout: An Analytic Approach", Ph.D. Dissertation, MIT Lab. Computer Science, Nov. 1980.

[Lee61] C. Y. Lee, "An Algorithm For Path Connection and its Applications", *IRE Transactions on Electronic Computers*, Vol. EC-10, 1961, pp. 346-365.

[Luk83] W. K. Luk, "A Greedy Switch-Box Router", *VLSI Document V158*, Carnegie-Mellon University, May 1983.

[Mat82] T. Matsuda, T. Fujita, K. Takamizawa, H. Mizumara, H. Nakamura, F. Kitajima, S. Goto, "LAMBDA: A Quick, Low Cost Layout Deisgn System for Master-Slice LSI's", *Proceedings of the 19th Design Automation Conference*, 1982, pp. 802-808.

[Meh84] S. Mehta, B. Kirk, M. Ng, R. Babbar, "CIPAR - A Complete Correct-by-Constration Placement and Routing System", *Proceedings of the 1984 IEEE Custom IC Conference*, 1984, pp. 117-121.

[Pre79] B. T. Preas, "Placement and Routing Algorithms for Hierarchical Circuit Layout", Ph.D. Dissertation, Department of Electrical Engineering, Stanford University, 1979.

[Riv82a] R. L. Rivest, C. M. Fiduccia, "A Greedy Channel Router", *Proceedings of the 19th Design Automation Conference*, 1982, pp. 418-424.

[Riv82b] "The 'PI' (Placement and Interconnect) System", *Proceedings of the 19th Design Automation Conference*, 1982, pp. 475-481.

[Sec84] C. Sechen, A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package", *Proceedings of the 1984 IEEE Custom IC Conference*, 1984, pp. 522-527.

[Sou81] J. Soukup, "Circuit Layout", *Proceedings of the IEEE*, Vol. 69, No. 10, Oct. 1981, pp. 1281-1304.

[Szy85] T. G. Szymanski, "Dogleg Channel Routing is NP-Complete", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-4, No. 1, Jan. 1985, pp. 31-41.

[Tin83] B. S. Ting, B. N. Tien, "Routing Techniques for Gate-Arrays", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No. 4, Oct. 1983, pp. 301-312.

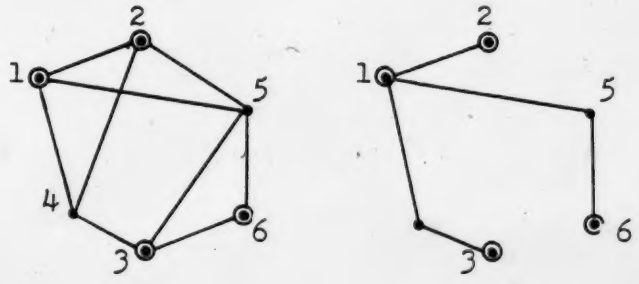
[Tsu83] S. Tsukiyama, I. Harada, M. Fukui, I. Shirakawa, "A New Global Router for Gate Array LSI", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No. 4, Oct. 83, pp. 313-321.

[Wie84] M. Wiesel, "Loose Routing For Gate Arrays", *Proceedings of the 1984 IEEE International Symposium on Circuits and Systems*, 1984, pp. 444-448.

[Yos82] T. Yoshimura, E. S. Kuh, "Efficient Algorithms For Channel Routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 1, Jan. 1982, pp. 25-35.

Appendix I - Steiner Tree in Graph Problem

Let $G=(V,E)$ denotes an undirected weighted graph where V is the set of vertices, E is the set of edges, and each edge is assigned a weight. Also let R be a subset of V. A *Steiner tree* T with respect to R in G is a tree in G that spans all vertices in R. Vertices in T that are not in R are called *Steiner points* or *Steiner vertices*. For example, Fig. A.1b shows a Steiner tree with respect to $R=\{1, 2, 3, 6\}$ in the graph of Fig. A.1a. A *minimum weight Steiner tree with respect to R in G* is defined to be any Steiner tree with respect to R in G where the total edge weight is minimum among all such trees. The problem of computing a minimum Steiner Tree is NP-complete [Gar79].



(a) Graph G(V,E)

(b) A Steiner Tree

Fig. A.1 Example of Steiner Tree

Appendix II - A Minimum-Spanning-Tree Global Router

The tree-type global routing algorithm we implemented is similar to the algorithms of [Aos83], [Sec84] and [Wie84]. Basically, the algorithm orders the nets by increasing number of terminals in the nets. In the sorted order, the algorithm finds, for each net, a minimum weight spanning tree of canonical wire segments which connect the net. The edge weight is $1/(2^{**}(D-d))$, where D=channel density, and d=maximum local density along the channel interval between the two terminals connected by the edge.

The minimum spanning tree is found using Kruskal's algorithm [Kru56]. Kruskal's algorithm takes $O(n\log_2 n)$ time to find the minimum spanning tree for a graph with n edges. Let N=number of nets, E=total number of canonical wire segments, e=average number of segments per net, i.e., $e=E/N$=average number of edges in the graph for each net. Therefore, on average, the time required to compute the minimum spanning tree for N nets each with e edges is about $O(Ne\log_2 e)=O(E\log_2 e)$ since $Ne=E$.